Trabajo Fin de Grado
Grado en Ingeniería Aeroespacial

Puesta a punto de un sistema de bajo coste para medir vibraciones estructurales

Autor: Paloma Monsalvete Contreras

Tutor: Pedro Galvín Barrera

Dpto. Mecánica de Medios Continuos y Teoría de Estructuras
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Grado
Grado en Ingeniería Aeroespacial

Puesta a punto de un sistema de bajo coste para medir vibraciones estructurales

Autor:
Paloma Monsalvete Contreras

Tutor:
Pedro Galvín Barrera
Catedrático de la Universidad de Sevilla

Dpto. Mecánica de Medios Continuos y Teoría de Estructuras
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Grado: Puesta a punto de un sistema de bajo coste para medir vibraciones estructurales

Autor: Paloma Monsalvete Contreras
Tutor: Pedro Galvín Barrera

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

A mi familia, a todas las personas que han hecho estos años un poco más sencillos.

Resumen

En este trabajo se ha desarrollado un sistema de medición de vibraciones estructurales de bajo coste. Para ello se ha utilizado una placa de Arduino UNO, junto con una IMU compuesta por tres acelerómetros y tres giróscopos.

Para que la monitorización se realice de forma inalámbrica, se ha añadido un módulo de radiofrecuencia, junto con un Arduino adicional y su correspondiente módulo de radiofrecuencia, que harán las veces de receptor.

Para comprobar la validez de dicho sistema completo, se realizaron mediciones a distintas frecuencias y amplitudes en el laboratorio del Departamento de Estructuras de la Universidad de Sevilla. Estas medidas se compararon con las tomadas por un acelerómetro de piezoeléctrico colocado sobre la misma superficie.

Respecto al postprocesado de datos, se ha hecho mediante código utilizando MATLAB, tanto en el caso del sistema de bajo coste como en el caso del sistema de piezoeléctrico. Es utilizando dicho software con el que se han conseguido además las gráficas aquí incluidas.

Abstract

In this work, a low-cost structural vibration measurement system has been developed. To do this, an Arduino UNO board has been used, together with an IMU composed of three accelerometers and three gyroscopes.

For the monitoring to be carried out wirelessly, a radio frequency module has been added, along with An additional Arduino and its corresponding radio frequency module, which will act as a receiver.

To verify the veracity of this complete system, measurements were made at different frequencies and amplitudes in the laboratory of the Department of Structures of the University of Seville. These measurements were compared with those taken by a piezo-electric accelerometer placed on the same surface.

Regarding the post-processing of data, it has been done through code using MATLAB, both in the case of the low-cost system and in the case of the piezo-electric system. It is by using said software that the graphs included here have also been obtained

Índice Abreviado

Resumen III
Abstract V
Índice Abreviado VII

1 Introducción **1**
1.1 Objeto y motivación del trabajo 3
1.2 Organización del trabajo 3

2 Hardware utilizado **5**
2.1 Arduino 5
2.2 MPU6050 8
2.3 Módulo NRF24L01 9
2.4 Protoboard 11
2.5 Presupuesto 11

3 Software utilizado **13**
3.1 Arduino IDE 13
3.2 Putty 15
3.3 MATLAB 17

4 Realización del proyecto **19**
4.1 Conexión de los dispositivos 19
4.2 Programación de los dispositivos 20

5 Ensayos realizados **27**
5.1 Montaje del sistema 27
5.2 Sistemas de referencia 27
5.3 Ensayos 29
5.4 Resultados obtenidos 29

6 Conclusiones **37**
6.1 Posibles desarrollos futuros 37

Apéndice A Códigos de Arduino **39**
A.1 Código de calibración 39
A.2 Código transmisor 40

A.3 Código receptor 42

Apéndice B Código de MATLAB 43

Apéndice C Montaje detallado 45

Índice de Figuras 49
Índice de Tablas 51
Índice de Códigos 53
Bibliografía 55
Glosario 57

Índice

Resumen III
Abstract V
Índice Abreviado VII

1 Introducción **1**
1.1 Objeto y motivación del trabajo 3
1.2 Organización del trabajo 3

2 Hardware utilizado **5**
2.1 Arduino 5
2.2 MPU6050 8
2.2.1 Comunicación I2C 9
2.3 Módulo NRF24L01 9
2.3.1 SPI 11
2.4 Protoboard 11
2.5 Presupuesto 11

3 Software utilizado **13**
3.1 Arduino IDE 13
3.2 Putty 15
3.3 MATLAB 17

4 Realización del proyecto **19**
4.1 Conexión de los dispositivos 19
4.2 Programación de los dispositivos 20
4.2.1 Calibración del sensor 20
4.2.2 Código del transmisor 20
4.2.3 Código del receptor 24
4.2.4 Código de postprocesado 24

5 Ensayos realizados **27**
5.1 Montaje del sistema 27
5.2 Sistemas de referencia 27
5.2.1 Acelerómetro de piezoeléctrico 27
5.2.2 BK Connect 29
5.3 Ensayos 29

5.4 Resultados obtenidos 29

6 Conclusiones 37
6.1 Posibles desarrollos futuros 37

Apéndice A Códigos de Arduino 39
A.1 Código de calibración 39
A.2 Código transmisor 40
A.3 Código receptor 42

Apéndice B Código de MATLAB 43

Apéndice C Montaje detallado 45

Índice de Figuras 49
Índice de Tablas 51
Índice de Códigos 53
Bibliografía 55
Glosario 57

1 Introducción

Es indispensable para el correcto desarrollo de una actividad ingenieril poner la seguridad del proyecto por encima de lo demás. Ya que si esto no fuese así, podrían producirse daños más allá de los materiales. No darle suficiente importancia a esto ha provocado diversas catástrofes a lo largo de los años. Es el caso del Hyatt Regency Hotel, en Kansas (Estados Unidos)[1], en el que dos pasarelas colgantes colapsaron debido a una sobrecarga de peso en las mismas. Esto se debió a un cambio en el diseño durante la construcción que no fue correctamente analizado y valorado, provocando así el que fue durante años el mayor accidente estructural de los Estados Unidos. Teniendo esto en cuenta, queda clara la importancia de realizar amplios estudios en las estructuras antes de su apertura al público para evitar desastres.

Dentro de dichos estudios, es importante el estudio de las vibraciones de una estructura, ya que también pueden ser la causa de diversos desastres. Esto fue lo que ocurrió en el caso del Tacoma Narrows Bridge, situado en Tacoma, Estados Unidos. El puente estuvo apenas cuatro meses en uso, antes de colapsar. Ya durante la construcción del mismo se detectaron problemas de vibraciones cuando soplava viento, aunque este no fuera necesariamente intenso, que motivaron la realización de diversos estudios en túneles de viento. El fallo se debió en líneas generales a las vibraciones estructurales que aparecieron por flameo aeroelástico, aunque realmente fue un conjunto de situaciones desafortunadas siendo el flameo una de las principales [2]. Por tanto, es importante realizar estudios precisos y rigurosos sobre las vibraciones estructurales que pueden soportar, ya que estas pueden acabar siendo las causantes de un fallo en una estructura.

Anteriormente, la falta de sistemas de precio asequible hacía necesario una gran inversión para

(a) El puente de Tacoma durante su innauguración [3].

(b) El puente de Tacoma tras colapsar.

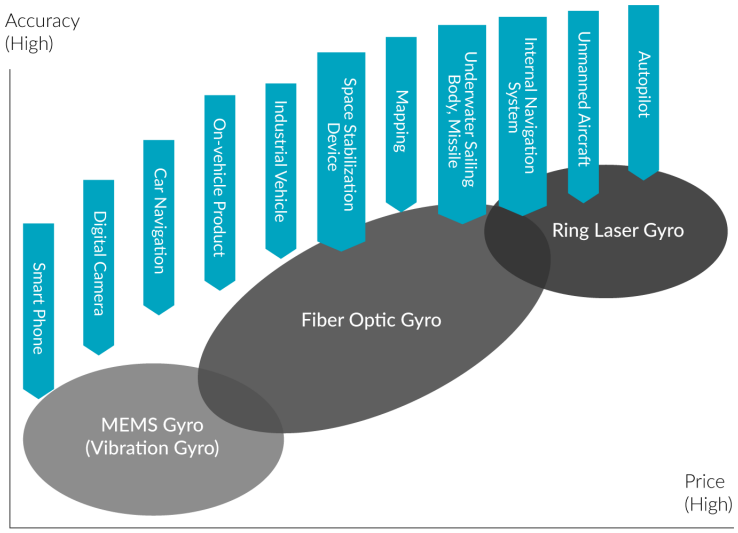


Figura 1.1 Tipos de giróscopos respecto a su precio [5].

realizar estudios ingenieriles, debido al alto precio de los equipos necesarios para ello. Sin embargo, en los últimos años la oferta de sistemas electrónicos de bajo coste ha ido aumentando, por lo que a día de hoy pueden encontrarse una gran variedad de los mismos. En general, estos sistemas llevan aparejada una pérdida de calidad en su desarrollo respecto a los sistemas de mayor coste, aunque esta pérdida no siempre es significativa. Además, la tecnología sigue avanzando actualmente, creando cada vez sistemas más eficaces.

Por tanto, puede decirse que los sistemas de bajo coste han recorrido un largo camino en su desarrollo hasta la actualidad. Un ejemplo de esto son los MEMS, sistemas de tamaño muy reducido que tienen en general bajo precio. Además, son sistemas precisos, rápidos y eficientes. Este tipo de sistemas han llegado a ser utilizados en aplicaciones profesionales como es el caso del vehículo espacial Soujourner, cuya misión en el año 1999 de casi tres meses de duración consistió en enviar datos sobre la superficie de Marte y cuyo diseño contaba con los giróscopos MEMS BEI GyroChip ([4]). Actualmente, el uso de sistemas micro-électro-mecánicos funciona como una parte fundamental de la filosofía More-Electric-Aircraft o Avión Más Eléctrico, que busca reducir el número de sistemas neumáticos y mecánicos de las aeronaves sustituyéndolos por sistemas eléctricos.

Ejemplos del interés en los sistemas electrónicos de bajo coste aplicados a diversos campos de estudio se presentan a continuación.En [6], A. Villarroel, G. Zurita y R. Velarde desarrollaron un sistema para medir las vibraciones de una máquina rotativa utilizando una placa de circuitos integrados basada en microcontroladores junto con un acelerómetro de piezoeléctrico de bajo coste y que enviaba los datos remotamente a un servidor utilizando protocolos TCP, además de guardarlos en una memoria SD. También centrado en el análisis de máquinas está [7], en el que se utiliza una placa similar a Arduino utilizando los esquemáticos proporcionados de forma libre por la empresa, llamada Freeduino, con acelerómetros MEMS y un módulo de radiofrecuencia de la marca XBee. En [8] se describe un sistema de bajo coste en el que se utilizan de nuevo componentes MEMS para medir las vibraciones de un vehículo, en lugar de equipos de precio superior para comprobar la funcionalidad del sistema, lo cual tendría multitud de finalidades como el estudio del estado de las carreteras.

También existen aplicaciones más centradas en análizar desastres naturales, por ejemplo, en [9] se

desarrolla un sistema de bajo coste que, mediante el uso de un sensor MEMS, sea capaz de detectar corrimientos de tierra de manera prematura, de forma que los daños producidos por estos fenómenos sean reducidos. En este último caso, se utiliza un microcontrolador, un sensor de vibración e impacto y un módulo WiFi para enviar los datos en tiempo real a largas distancias.

En el campo del estudio de estructuras, como es el objeto de este trabajo, se encuentran diversos estudios similares. Una de ellas es el estudio de las vibraciones producidas en un puente, como es el caso de [10] en el que se estudian las vibraciones a las que se encuentra sometido un puente sobre el río Pisuerga en Valladolid, comparando los datos obtenidos por distintos acelerómetros MEMS en comparación con acelerómetros de piezoeléctrico convencionales. También se utilizan esos sistemas para estudiar y monitorizar estructuras y edificios históricos [11].

1.1 Objeto y motivación del trabajo

El objeto de este trabajo es diseñar un sistema de bajo coste, utilizando una placa de circuitos integrados basada en microcontroladores y una IMU basada en sistemas MEMS, que sea capaz de monitorizar y medir con suficiente precisión las vibraciones de una estructura. La placa elegida es el dispositivo Arduino UNO y la Unidad de Medidas Inerciales, el módulo MPU-6050. Ambos son sistemas muy utilizados y que cuentan con un precio muy reducido, respecto a lo que sería el equipo estándar. Además, se incluye en el equipo un módulo transceptor, el NRF24L01, cuya función será enviar los datos en tiempo real a otra placa que tenga conectado otro módulo del mismo tipo, de forma que estos se reciban en un ordenador durante los ensayos.

La motivación que ha llevado a desarrollar el trabajo aquí presente es comprobar si es posible que un sistema de bajo coste como el que aquí se describe, cuente con las prestaciones necesarias para realizar dichos ensayos y conseguir los requisitos de precisión. Como se ha comentado anteriormente, los sistemas de bajo coste y los sistemas micro-electro-mecánicos han avanzado considerablemente desde que aparecieron en la industria, por lo que, considerando la diferencia de precio con los sistemas estándar, es natural comprobar la validez de dichos equipos en este campo de estudio. Además, existía un interés personal en profundizar en los sistemas de Arduino, tras haber realizado breves desarrollos a lo largo del grado.

1.2 Organización del trabajo

El trabajo está dividido en distintas secciones. En la sección 2 se explican detalladamente los sistemas que se han utilizado en el desarrollo del mismo, tanto la placa de circuitos integrados como los módulos que se han conectado a ella. Además, se incluye una introducción sobre Arduino y sus distintos dispositivos.

En la sección 3 se puede encontrar una descripción de los programas que han sido necesarios. Dentro de estos programas encontramos el IDE de Arduino, Putty, que ha sido utilizado como monitor serial para recibir los datos durante los ensayos, y MATLAB, que ha sido el programa elegido para realizar el análisis y procesado de los datos.

Posteriormente, en la sección 4 se ha descrito paso a paso el desarrollo del proyecto y la puesta a punto de los equipos. También pueden encontrarse los diagramas de flujo de los códigos utilizados para programar los dispositivos.

En el capítulo 5 se describen los ensayos realizados y los sistemas utilizados como referencia, en este caso, el acelerómetro de piezoeléctrico y su correspondiente software. En este mismo capítulo, se encuentra una sección (Sección 5.4), en la que se presentan los resultados obtenidos durante los ensayos realizados. Los resultados de cada ensayo se presentan en dos gráficas, una correspondiente al dominio del tiempo, y otra correspondiente al dominio de la frecuencia.

Por último, en la sección 6 se desarrollan una serie de conclusiones a las que se ha llegado tras la realización de este trabajo y se presentan algunas líneas posibles de desarrollos futuros.

Al final del trabajo se incluyen una serie de anexos. Primero, están descritos los códigos utilizados para programar las dos placas de Arduino utilizadas, escritos en lenguaje C, y los códigos de MATLAB correspondientes al análisis y procesado de los datos tras los ensayos. Finalmente, se incluye un anexo en el que se explica la puesta a punto del equipo de Arduino paso a paso.

2 Hardware utilizado

2.1 Arduino

Arduino nace en el año 2005 en el Ivrea Interaction Design Institute, Ivrea (Italia) [12], gracias a la necesidad de los profesores universitarios de contar con un sistema de bajo coste para el desarrollo de sus proyectos . Hasta este momento, comprar placas de circuitos integrados basadas en microcontroladores conllevaba un gasto elevado, lo cual hacía difícil el desarrollo de distintos proyectos en la Universidad si no se contaba con un alto presupuesto. Desde ese momento, han aparecido distintas opciones dentro del sector de sistemas electrónicos de bajo coste, aunque Arduino sigue siendo una de las más utilizadas.

Un Arduino es una placa hardware de circuito integrado que se programa mediante instrucciones de código para que realize lo que el usuario desea. Es de código abierto, lo que permite que cualquier usuario pueda programarla. En la placa se incluye todo lo necesario para poder ponerla a punto y la gran variedad de módulos y accesorios disponibles hacen que sea muy versátil y permita realizar una gran cantidad de proyectos. Tiene desde aplicaciones meramente académicas, como este trabajo y otros muchos como [13] y [14] que se pueden encontrar en el depósito de trabajos de la Universidad, hasta aplicaciones más cotidianas. Un ejemplo de esto último sería el uso que se hace de Arduino para domótica en los hogares..

Una de las grandes ventajas de Arduino es que, como ya se ha comentado, es un sistema de código abierto. Además, al estar su uso tan extendido, existen gran cantidad de foros y comunidades en las que los usuarios colaboran unos con otros en proyectos. De esta manera se crea una retroalimentación entre los distintos usuarios tratando de buscar soluciones a los problemas que se presentan. Existen gran cantidad de librerías con funciones ya creadas para distintos módulos que se encuentran en el mercado. Estas librerías son gratuitas y pueden ser tanto librerías oficiales de Arduino, como librerías escritas por otros usuarios que se encuentran en páginas web como GitHub

Figura 2.1 Logo Arduino.CC.

o GitLab.

Dentro de Arduino, existen distintos modelos de placas. El más pequeño es el Arduino Nano Every, que cuenta con una superficie de, aproximadamente, $8.1cm^2$ y es el más barato de la colección con un precio de 9€ en la web oficial. Por el otro lado, el modelo más grande es el Arduino MEGA, que tiene una superficie aproximada de $54.1cm^2$. Este último cuenta con un total de 54 pines digitales, 16 pines analógicos y 4 puertos seriales, lo cual implica que el aumento de tamaño conlleva un aumento de la capacidad de la placa. Además, la placa Arduino MEGA cuenta con un microcontrolador más potente, lo que hace que sea la óptima en el caso de proyectos grandes que necesiten una gran capacidad de procesamiento para funcionar.

Teniendo todo esto en cuenta, no existe una placa de Arduino que sea esencialmente mejor que la otra, sino que esto dependerá de la función que se desee realizar y de una gran cantidad de elementos como el espacio disponible o el presupuesto.

El modelo que se tratará en el trabajo aquí presente es Arduino UNO. Es el modelo más utilizado, sobretodo a nivel académico, ya que no siempre es necesario utilizar la placa más potente. En este caso, no se necesita un microcontrolador de gran capacidad, debido a que se conectará un número pequeño de módulos a cada placa (dos y uno, respectivamente) y no son módulos que necesiten demasiada potencia para funcionar correctamente. Por todo esto, el Arduino UNO es capaz de cumplir los requerimientos del proyecto. Este modelo puede comprarse por sí solo o bien como parte de packs de iniciación que incluyen una gran cantidad de elementos como diodos LED, resistencias, pantallas LCD, etc. para poder realizar distintos proyectos.

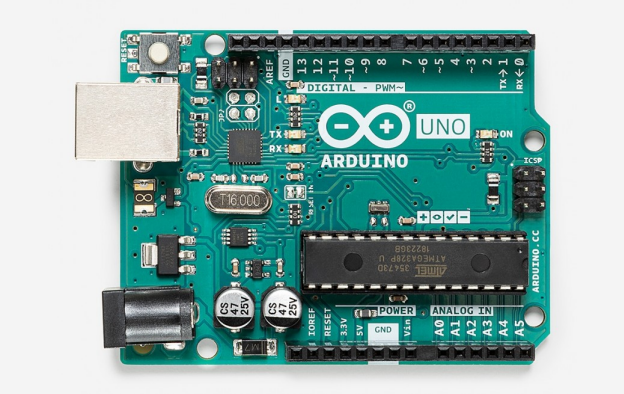


Figura 2.2 Placa Arduino UNO.

Características de Arduino UNO [12]:

- Microcontrolador: ATMEGA328P.
- Voltaje de funcionamiento: 5V.
- Voltaje de entrada recomendado: 7-12V.
- Cuenta con 6 pines analógicos, que funcionan únicamente como entradas. Estos vienen marcados en la placa desde A0 hasta A5.
- Cuenta con 14 pines digitales de entrada/salida. Vienen numerados desde el 0 hasta el 13. Además, entre estos 14 pines, 6 de ellos pueden manejar señales PWM. Estos últimos vienen definidos con el símbolo ~ al lado de su correspondiente número.
- Además de los pines ya comentados, dispone de un puerto USB A y un puerto jack.

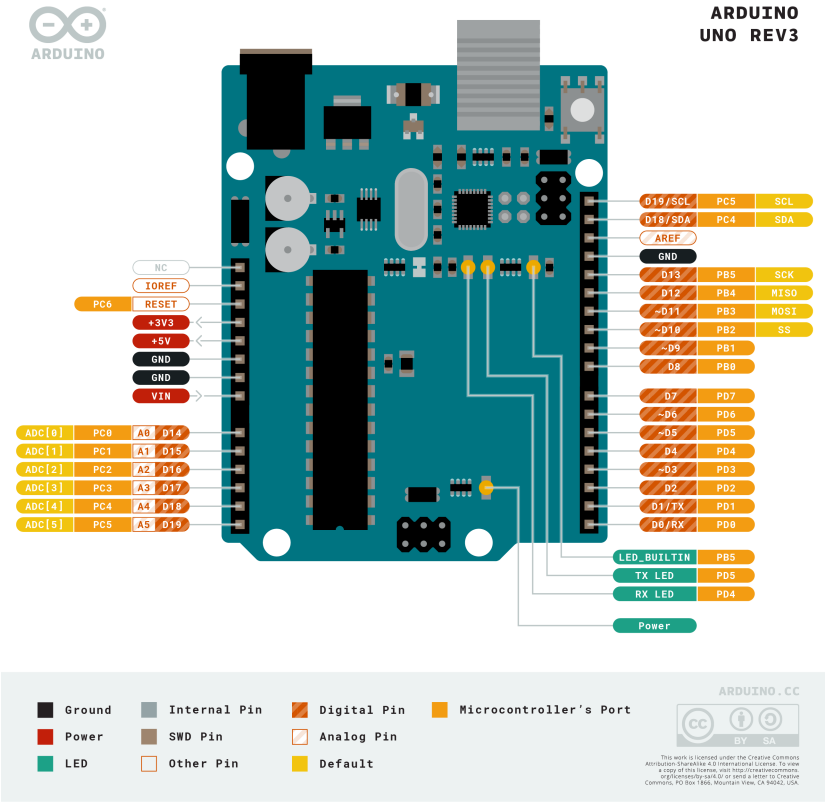


Figura 2.3 Diagrama de pines de la placa Arduino UNO.

- Memoria Flash: 32 KB.
- Memoria RAM: 2 kB
- Memoria EEPROM: 1 kB
- Reloj de cuarzo de 16 MHz.
- Dimensiones: 68.6 mm x 53.4 mm.
- Peso: 25 g.
- Precio: 20€, en la página web de ArduinoCC.

Además de ser un sistema de bajo coste, con un precio significantemente menor que los de otros sistemas similares, Arduino pone a disposición de los usuarios los esquemáticos de sus productos, para que el usuario tenga la posibilidad de crear su propia placa desde cero.

Las placas Arduino pueden funcionar tanto conectadas a un ordenador mediante un cable de USB A a USB B, como de forma autónoma. Para esto último existen diversas opciones. Pueden utilizarse portapilas, baterías o transformadores que se conecten a la corriente eléctrica. Estos sistemas pueden ir conecctados o bien al puerto jack de la placa, al puerto USB A hembra, o al pin V_{in} utilizando cables de conexión de tipo macho.

Existe además un modelo especial dentro del Arduino UNO, que es el Arduino UNO WiFi. Este tiene características muy similares al aquí explicado pero con mejores prestaciones. La principal diferencia reside en que esta placa cuenta con la posibilidad de conectarse a WiFi o Bluetooth sin necesidad de utilizar módulos adicionales para ello. Aun así, la placa tiene un precio más elevado que el modelo convencional.

Tabla 2.1 Precisión del sensor MPU6050.

Función	Precisión
Acelerómetro	2g,4g,8g y 16g
Giróscopo	250,500,1000 y 2000º/s

2.2 MPU6050

El módulo MPU-6050 [15] es una IMU de seis grados de libertad. Cuenta, por tanto, con tres acelerómetros y tres giróscopos, de modo que nos permite medir tanto aceleraciones lineales como velocidades angulares. Es un sensor muy económico, lo que hace que sea uno de los más utilizados. El sensor viene montado en un módulo, de forma que se facilita su uso, haciendo más accesibles los pines desde el exterior. En el caso de este trabajo, sin embargo, va a utilizarse únicamente uno de los grados de libertad del acelerómetro, aunque es fácilmente extensible al resto. El giróscopo, sin embargo, no es necesario para este proyecto.

Figura 2.4 Módulo MPU6050.

La alimentación se realiza entre 2.4V y 3.6V. Aunque en este caso, una de las ventajas brindadas por el módulo GY-521 es que este incluye un regulador de tensión, de forma que puede alimentarse conectándolo al pin de 5V de la placa. El módulo cuenta con un diodo LED de color verde que se enciende para indicar que el este tiene corriente.

Respecto a la precisión del sensor, esta puede modificarse según las necesidades del proyecto a realizar. Las precisiones disponibles pueden verse en la tabla 2.1. La precisión por defecto son 2g, aunque puede modificarse con líneas de código. En este proyecto seha mantenido a 2g, debido a que es la que mejor se adapta a las necesidades del mismo.

Se comunica utilizando protocolos I2C, y soporta una velocidad de hasta 400Hz. Es por este motivo por el cual es necesario utilizar las librerías de comunicacion I2C [16] cuando se programe este módulo, como puede verse en el código correspondiente al anexo A.2.

Dentro de la librería [17] del sensor, se encuentran una variedad de funciones. Las funciones utilizadas en el desarrollo de este trabajo se detallan a continuación:

- initialize(): Permite inicializar el sensor.

- testConection(): Devuelve 1 en caso de que la conexión se haya producido correctamente, y un 0 en caso contrario.
- getAcceleration(ax,ay,az): Guarda en las variables ax, ay y az los valores de las aceleraciones. Es importante tener en cuenta que estos son valores "raw" y, por tanto, hay que pasarlos a Sistema Internacional.
- getAccelerationX(ax): Guarda en la variable ax el valor de la aceleración sobre el eje X.
- getAccelerationY(ay): Guarda en la variable ax el valor de la aceleración sobre el eje Y.
- getAccelerationZ(az): Guarda en la variable ax el valor de la aceleración sobre el eje Z.
- getXAccelOffset(): Devuelve el valor del offset sobre el eje X.
- setXAccelOffset(ax_o): Establece el valor de ax_o como offset sobre el eje X.
- getYAccelOffset(): Devuelve el valor del offset sobre el eje Y.
- setYAccelOffset(ay_o): Establece el valor de ay_o como offset sobre el eje Y.
- getZAccelOffset(): Devuelve el valor del offset sobre el eje Z.
- setZAccelOffset(az_o): Establece el valor de ax_o como offset sobre el eje Z.

Estas nos permiten obtener fácilmente los valores que están tomando las aceleraciones y los ángulos de giro, obtener los Offset que tiene el sensor programados o cambiar estos valores, entra otras muchas funciones. Esto último es importante, ya que en [18] se describe la importancia de que los sensores de un proyecto estén bien calibrados, y más aún, si se van a tratar valores pequeños, como es el caso de este trabajo.

2.2.1 Comunicación I2C

La comunicación I2C fue introducida en el mercado en 1982. Es un protocolo de comunicación de información entre circuitos integrados. Una de sus grandes ventajas es que necesita solo dos señales y, por tanto, dos cables; uno para la señal de reloj (SCL) y otro para la señal de datos (SDA). El sistema utiliza una estructura Maestro-Esclavo. El dispositivo que trabaja como Maestro es el encargado de la señal de reloj, que es única para ambos equipos, de forma que es más sencillo que ambos se encuentren sincronizados. Además, es también el maestro el que inicia y detiene la comunicación entre ambos dispositivos. Por su parte, el dispositivo que trabaja como Esclavo es el encargado de proporcionar la información de interés.

En este caso, el dispositivo maestro es el Arduino UNO, mientras que el esclavo en el sensor MPU-6050.

2.3 Módulo NRF24L01

Para enviar los datos en vivo desde la placa transmisora, colocada en la estructura que se desee medir, hasta la placa receptora, conectada al ordenador, se utilizará el módulo NRF24L01 [19].

Dicho módulo cuenta con un transceptor de radiofrecuencia , de forma que puede funcionar tanto de transmisor como de receptor y que trabaja entre 2400MHz y 2525MHz (divididos en 125 canales de 1MHz cada uno).

Existen dos modelos similares de este mismo módulo. El modelo más básico, que es el utilizado en este proyecto, que se puede ver en la figura 2.5, y un módelo que cuenta con una antena externa, lo cual aumenta su alcance significativamente. En el caso del modelo básico, cuya antena

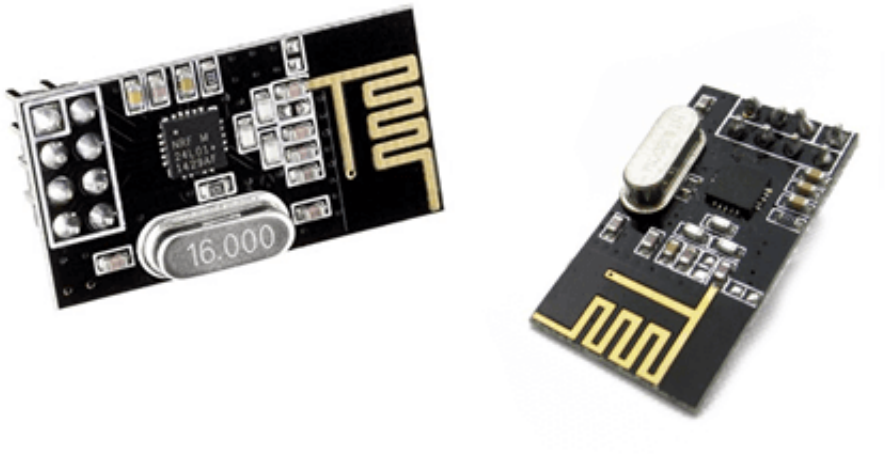


Figura 2.5 Módulo NRF24L01.

está integrada en el módulo, el alcance teórico es de 100 metros. Sin embargo, el alcance real es significativamente menor. El módulo utiliza protocolos de comunicación SPI, por lo que es necesario incluir estas librerías en el código correspondiente.

Se alimenta entre 1.9 y 3.6 V, por lo que es importante conectarlo al pin de 3.3 V de salida, ya que en caso de conectarlo a 5V podría dejar de funcionar correctamente. Mientras está trabajando como transmisor tiene un consumo aproximado de 11.3mA, mientras que funcionando como receptor, su consumo aproximado es 12.3mA.

El módulo permite una conexión estable entre los distintos dispositivos. En este caso se va a conectar únicamente a un dispositivo, pero cabe descatar que el NRF24L01 permite conectarse simultáneamente a seis de ellos.

Al igual que en el caso del módulo MPU-6050, se utilizará la librería correspondiente a este componente. A continuación se detallan algunas de las funciones de esta librería.

- openWritingPipe(pipe): Es la función encargada de comenzar la comunicación en la dirección que indice la variable pipe. Esta función se debe escribir en el código correspondiente al transmisor.
- write(data, sizeof data): El dispositivo envía al receptor la información que se encuentre en la variable data.
- openReadingPipe(pipe):Es la función encargada de comenzar la comunicación en la dirección que indice la variable pipe. Esta función se debe escribir en el código correspondiente al receptor.
- StartListening(): El dispositivo comienza a recibir datos.
- read(data, sizeof data): El dispotivo recibe la información guardada en la variable data.

2.3.1 SPI

Este es otro de los protocolos de comunicación más utilizados, junto con el I2C explicado anteriormente. En este caso, son necesarios cuatro cables: uno para la señal de reloj (SCL), uno para enviar datos del Maestro al Esclavo (MISO), uno para enviar datos del Esclavo al Maestro (MOSI) y una última señal para seleccionar el esclavo deseado. Esto último es necesario porque el protocolo SPI permite trabajar con varios esclavos la mismo tiempo. De nuevo, al igual que en el caso anterior, la señal de reloj es fundamental para mantener la sincronización entre los dispositivos, y esta es controlada por el dispositivo que funciona como Maestro.

El maestro, en este caso, es la placa de Arduino UNO, mientras que el esclavo es el módulo NRF24L01.

2.4 Protoboard

Para conectar todo el sistema es necesaria una placa de pruebas o protoboard, como la de la imagen 2.6. Estas placas son muy utilizadas en ámbitos electrónicos.

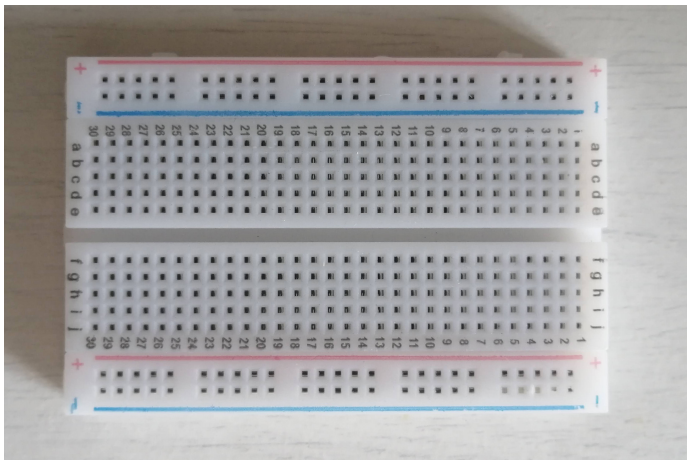


Figura 2.6 Placa de pruebas.

Su estructura exterior es de plástico, sin embargo, en su interior, cuenta con líneas metálicas que conectan los distintos orificios de la placa. Las conexiones siguen el patrón expuesto en la figura 2.7. En dicho esquema se puede ver que las perforaciones se encuentran conectadas entre ellas en el mismo sentido, excepto en el caso de las líneas de los extremos, representadas en rojo y negro en el esquema, que están especialmente pensadas para los polos positivo y negativos de la fuente de tensión. Son componentes muy asequibles y que facilitan en gran medida la conexión de sistemas, ya que no es necesario soldar pines a los componentes para utilizarlo, basta con cables de conexión de tipo macho.

2.5 Presupuesto

En la tabla 2.2 puede verse un presupuesto aproximado de los dispositivos aquí utilizados, a modo ilustrativo.

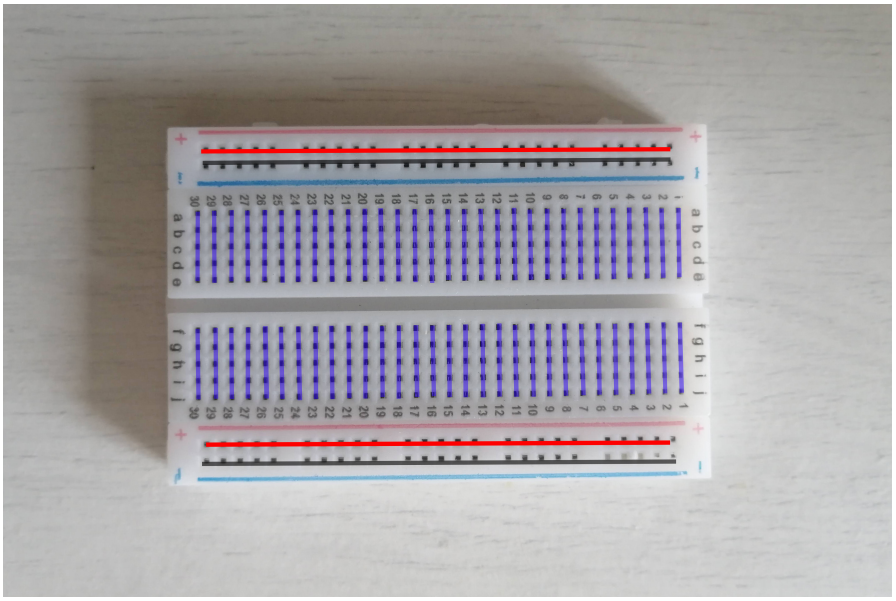


Figura 2.7 Conexiones en una placa de pruebas.

Tabla 2.2 Presupuesto aproximado del sistema.

Dispositivo	Cantidad	Precio unitario (€)	Precio total (€)
Arduino UNO	2	20	40
MPU6050	1	2.64	2.64
NRF24L01	2	2.83	5.66
Cable USB-A a USB-B	1	2.70	2.70
Fuente de alimentación 9 V	1	3.07	3.07
Paquete de cables para protoboard	1	1.90	1.90
Protoboard	1	0.97	0.97
Total			56.94

3 Software utilizado

3.1 Arduino IDE

Para programar el código de nuestro proyecto, Arduino facilita un software llamado IDE de forma gratuita. Este puede descargarse fácilmente de la web de Arduino.

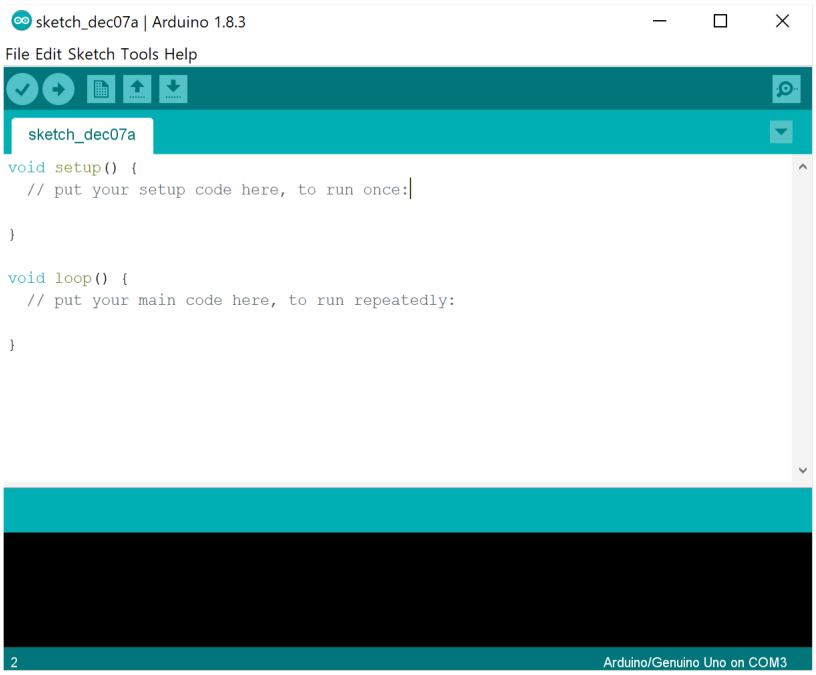


Figura 3.1 IDE.

El IDE nos permite, por una parte, programar de manera sencilla, sin necesidad de una conexión a Internet durante el proceso. Además, es posible acceder a las distintas librerías con códigos de ejemplo, que facilitan la tarea. Por último, permite la compilación del programa, y la subida a la placa de manera inmediata.

Tal y como se ve en la figura 3.1, en la ventana principal encontramos un editor de texto, una ventana de mensajes, en la que se avisa al usuario en caso de algún fallo en la compilación o en el envío del programa que corresponde a la sección de fondo negro de la parte inferior, y, en la parte superior, distintos iconos de opciones y los menús desplegables, que se pueden ver con más detalle

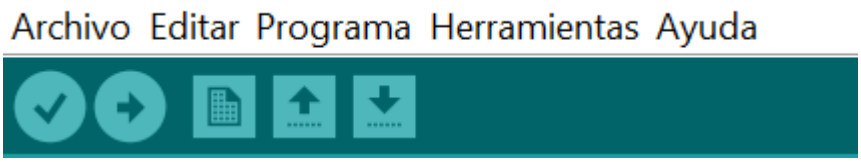


Figura 3.2 Opciones disponibles en IDE.

en la imagen 3.2. El primero de estos iconos, un tick, corresponde a la compilación. El icono de la flecha horizontal representa la subida a la placa del programa en cuestión. Cuando pulsamos este, antes de enviar el programa, se compila automáticamente y, en caso de error, no se envía y se muestra el error por pantalla. Los últimos tres iconos corresponden a "Nuevo", "Abrir" y "Guardar".

Respecto al lenguaje de programación, otra de las grandes ventajas del sistema es que utiliza un lenguaje de alto nivel, de modo que es sencillo de aprender y de escribir. Además, en la propia web de Arduino, se proporciona ayuda sobre las funciones e instrucciones disponibles. El lenguaje utilizado está basado en C++, aunque cuenta con algunos detalles en los que difieren ambos lenguajes.

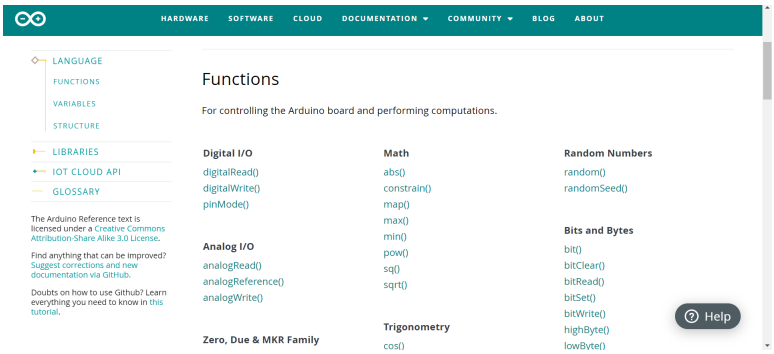
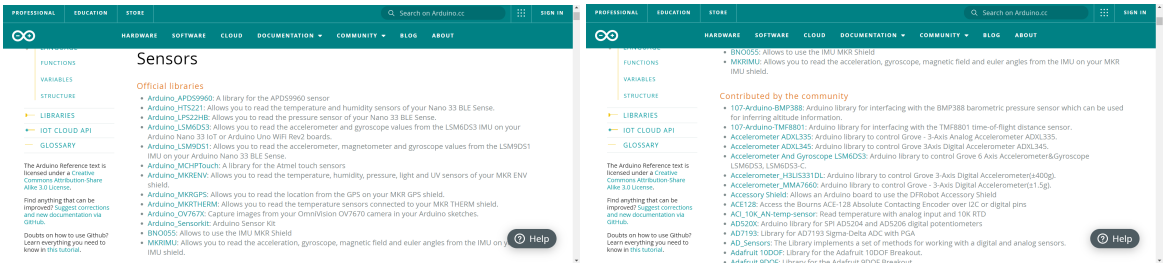


Figura 3.3 Documentación sobre las funciones disponibles en la web de Arduino.

También están disponibles en dicha página, una gran colección de librerías de distintos componentes, como son [16], [17] y [20]. Dentro de dichas librerías, encontramos tanto librerías proporcionadas por la propia empresa como librerías aportadas por los usuarios de manera gratuita. Estas facilitan la tarea de programación, ya que, una vez instaladas, permiten realizar numerosas funciones utilizando pocas líneas de código. Además, con las librerías vienen incluidos una serie de ejemplos de código, a los que se accede desde la pestaña "File".

En 2019 se publicó una versión superior, llamada IDE 2.0, con nuevas funcionalidades. También



(a) Librerías oficiales. **(b)** Librerías aportadas por los usuarios.

Figura 3.4 Librerías.

está disponible un editor web, llamado Arduino Web Editor, muy similar al IDE original pero con las ventajas que conlleva que este se encuentre conectado a la nube y, por tanto, permita el acceso a los programas desde otros dispositivos.

Entre las distintas funciones de este entorno integrado se encuentra la posibilidad de ver en pantalla los datos que se reciben por el Puerto Serie. Además, también permite la ópción de ver una gráfica con dichos datos en tiempo real, si pulsamos en la opción "Tools" y "Serial Plotter". El problema de estas dos opciones, es que no permiten guardar dichos resultados en un archivo de manera automática, motivo por el cual no se han utilizado en este trabajo.

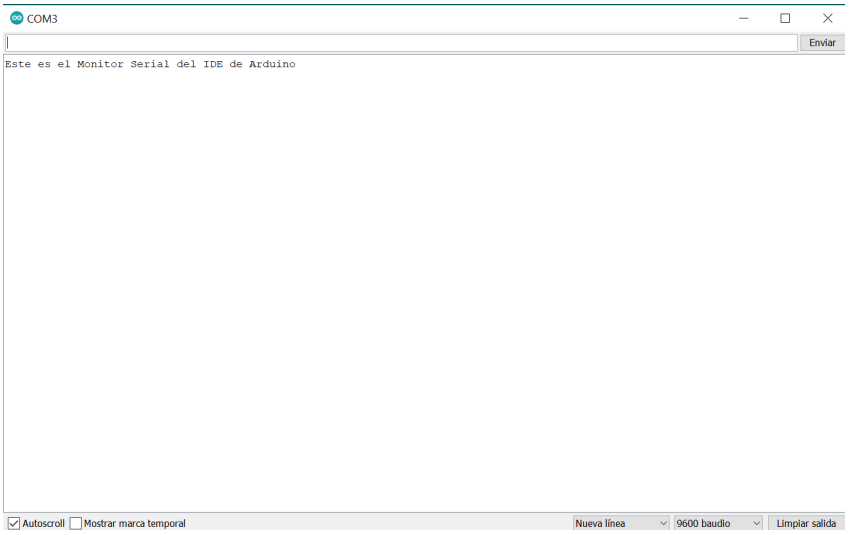


Figura 3.5 Monitor serial del IDE.

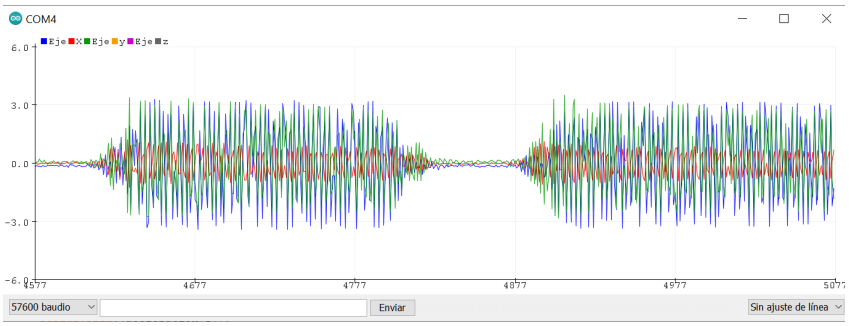
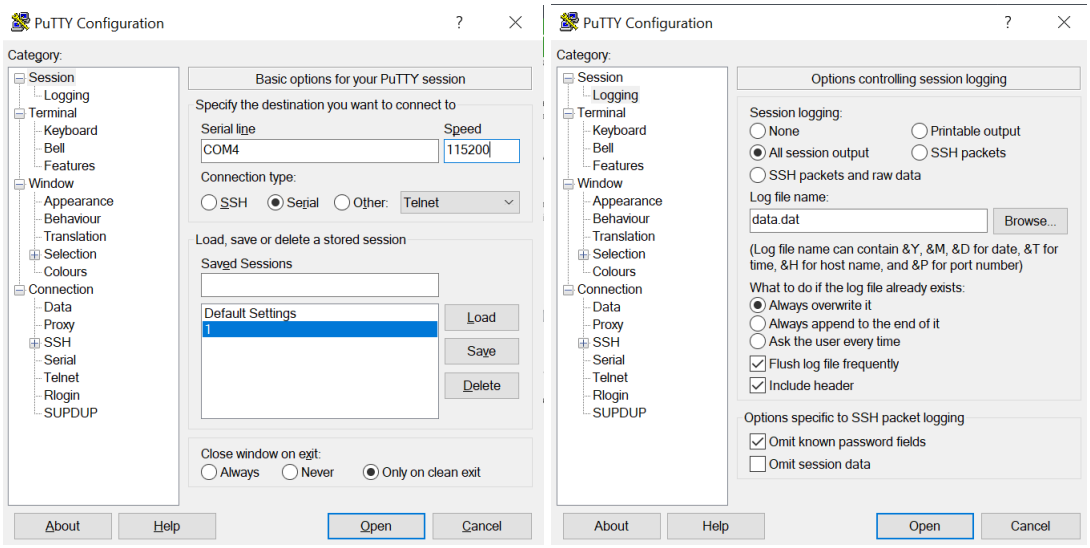


Figura 3.6 Serial Plotter del IDE de Arduino.

3.2 Putty

Tal y como se ha explicado en la sección anterior, el monitor serial que incorpora el entorno integrado de Arduino no permite guardar de manera automática los archivos en el ordenador. Aunque esto no supone un problema importante, teniendo en cuenta la cantidad de muestras tomadas, sí que resultaría conveniente poder guardar el archivo automáticamente. Tras investigar distintas opciones viables, se optó por el programa Putty. Este programa utiliza protocolos SSH y TelNet para comunicarse de manera remota con otros dispositivos y permite visualizar los datos recibidos por puerto serie.



(a) (b)

Figura 3.7 Configuración de Putty.

Figura 3.8 Monitor serial de Putty.

Putty es un programa que fue creado en 1997 por Simon Tathan, y que a día de hoy funciona gracias a su creador y a un equipo de voluntarios [21]. Es de código libre, igual que el caso del IDE, y puede descargarse gratuitamente de su página web.

En el caso de este trabajo, se utilizó como monitor serial, guardando los datos recibidos en un archivo .dat en el directorio indicado. Para ello basta con configurarlo con las opciones que pueden verse en la figura 3.7. Al cerrar la ventana del monitor serial, el archivo queda guardado en la ruta indicada en la configuración.

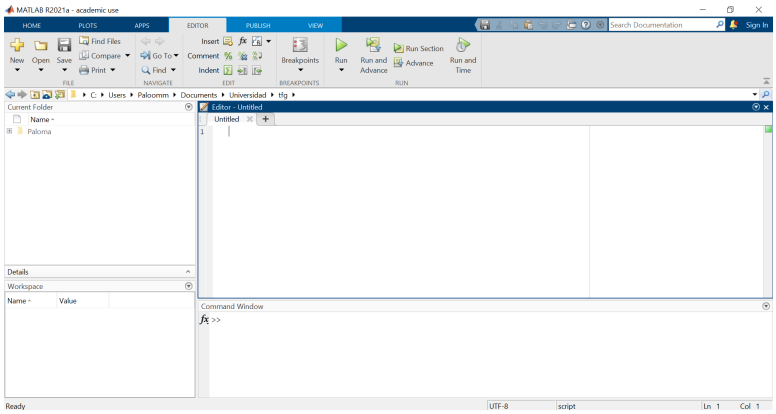


Figura 3.9 Pantalla principal de MATLAB.

3.3 MATLAB

Finalmente, para realizar el postprocesado de los datos, se utilizó MATLAB [22]. Este es un programa principalmente de cálculo numérico, aunque a día de hoy cuenta con un gran número de opciones y posibilidades. Es un programa no gratuito, es decir, se necesita una licencia de pago para que funcione.

En este caso, se utilizó para leer los datos de los archivos correspondientes, analizarlos y, por último, realizar las representaciones gráficas que se encuentran en capítulos posteriores.

Aunque en este trabajo no se ha utilizado, MATLAB permite la opción de programar Arduino y reconoce la placa cuando esta está conectada. Por tanto, habría sido otra opción en caso de no poder o no querer utilizar el IDE para programarla.

4 Realización del proyecto

4.1 Conexión de los dispositivos

El esquema general del proyecto puede verse en la figura 4.1 . En dicha figura, la línea discontinua representa los datos de aceleración que se envían desde el Arduino transmisor hasta el Arduino receptor. Para conectar los módulos a las placas correspondientes se ha seguido el esquema de las

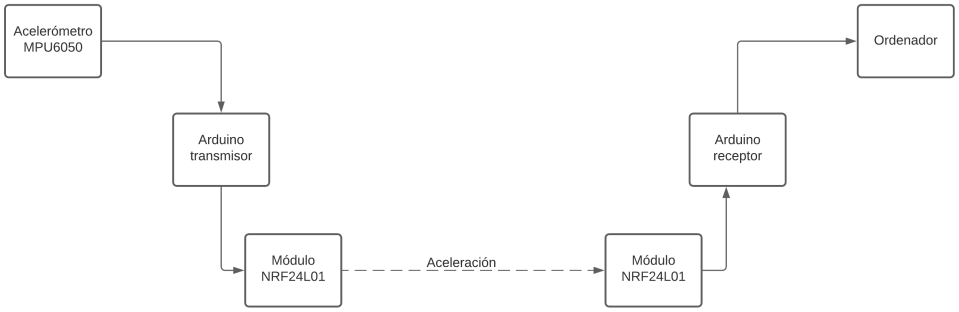


Figura 4.1 Esquema de las conexiones del sistema..

tablas 4.1 y 4.2.

El módulo MPU6050 queda conectado por tanto a los pines analógicos del Arduino, además de a su correspondiente alimentación y tierra. Para realizar estas conexiones existía la posibilidad de soldar los pines al módulo mediante un soldador de estaño o utilizar cables macho/macho, en las dos opciones el módulo quedaba conectado a una placa de pruebas o protoboard que, a su vez, se conectaba a la placa Arduino. Ambas opciones son similares así que, por simplicidad, se utilizaron cables.

Además de estas conexiones es necesario mencionar la alimentación de las placas. El Arduino receptor queda alimentado por el ordenador, al estar conectado a este mediante USB, con un voltaje de 5 V. Para alimentar al Arduino transmisor, que tiene que funcionar sin una conexión al ordenador, ya que este se encontrará sobre la superficie sobre la que se desee medir; se valoraron distintas opciones como utilizar una batería como la de la imagen 4.2a o un transformador para conectarlo a la corriente eléctrica. Finalmente, se optó por esta última opción.

Tabla 4.1 Conexión del acelerómetro MPU6050 con Arduino.

Sensor MPU6050	Arduino UNO
Vcc	5V
GND	GND
SCL	A5
SDA	A4
XOA	Sin conectar
XCL	Sin conectar
AD0	Sin conectar
INT	Sin conectar

Tabla 4.2 Conexión del módulo NRF24L01 con Arduino.

Módulo NRF24L01	Arduino UNO
Vcc	3.3V
GND	GND
CE	D9
SCN	D10
MOSI	D11
MISO	D12
SCK	D13
IRQ	Sin conectar

4.2 Programación de los dispositivos

4.2.1 Calibración del sensor

El primer paso es calibrar el sensor. Esto es, tratar de que las medidas sean cero en el momento en el que la superficie no se encuentre vibrando. Para ello es importante colocar la placa sobre la superficie que se va a medir y no moverla hasta que la calibración haya terminado. Este paso es especialmente importante ya que se trabajará con valores de aceleración pequeños.

Este código se ejecuta con la placa conectada al ordenador, sin necesidad de utilizar el módulo NRF24L01. Es por esto por lo que en el código se incluyen únicamente las librerías correspondientes al módulo MPU6050. Primeramente, se reciben del dispositivo los Offsets que en ese momento están guardados en el sensor. El código consiste en una serie de bucles iterativos consecutivos, de forma que finalizará en el momento en que todos los ejes estén ajustados. Para realizar el ajuste se modifican estos Offsets hasta que se consiga un valor nulo o cercano a cero en las aceleraciones de los ejes. El código está disponible en el anexo A.1 y su diagrama de flujo puede verse a continuación, en la figura 4.3.

4.2.2 Código del transmisor

Para programar esta placa es necesario tener en cuenta que tiene dos funciones simultáneas. Por un lado, tiene que leer los datos del sensor de aceleración y, por otro, mediante el módulo de radiofrecuencia, enviar los datos hasta la placa que se encuentra conectada al ordenador. Para realizar estas funciones, se han utilizado las librerías correspondientes a ambos módulos, que aparecen en los

(a) **(b)**

Figura 4.2 Posibilidades valoradas para alimentar el Arduino en remoto.

códigos descritos en los anexos.

Cabe destacar la importancia de pasar los datos a Sistema Internacional. El sensor cuenta con una resolución de 16 bits, por tanto, los valores en bruto de sus medidas serán desde -32768 hasta 32767. Será necesario entonces escalar los valores mediante el uso de factores de conversión. Para ello es necesario recordar que la precisión del sensor puede tomar distintos valores, que son 2g, 4g, 8g y 16g. Utilizando la fórmula 4.1 se obtienen los valores en Sistema Internacional. Si se desean obtener las medidas en g(un g equivale a 9.81 m/s^2), en vez de utilizar la fórmula anterior, se utilizará la fórmula 4.2.

En el código en cuestión se ha utilizado únicamente el valor de la aceleración en el eje Z, ya que el acelerómetro de piezoeléctrico con el que se comparan los valores es un acelerómetro de un solo eje, al contrario que el MPU-6050, que puede obtener las medidas de tres ejes.

El código completo puede verse en el apéndice A.2. Por su parte, el diagrama de flujo de dicho código queda definido en la figura 4.4.

$$acc_{SI} = \frac{2 \times 9.81}{32768} \tag{4.1}$$

$$acc_g = \frac{2}{32768} \tag{4.2}$$

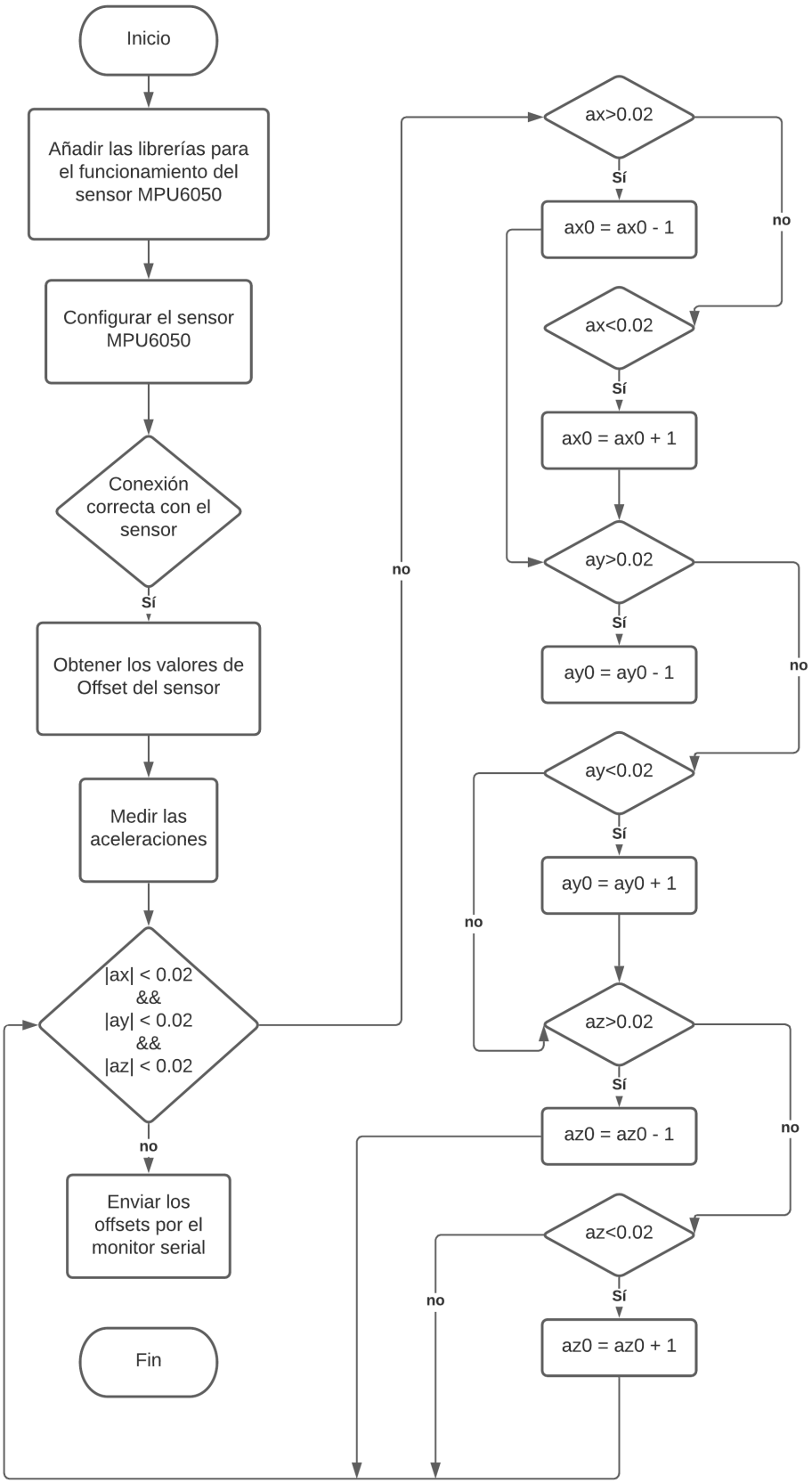


Figura 4.3 Diagrama de flujo de la calibración.

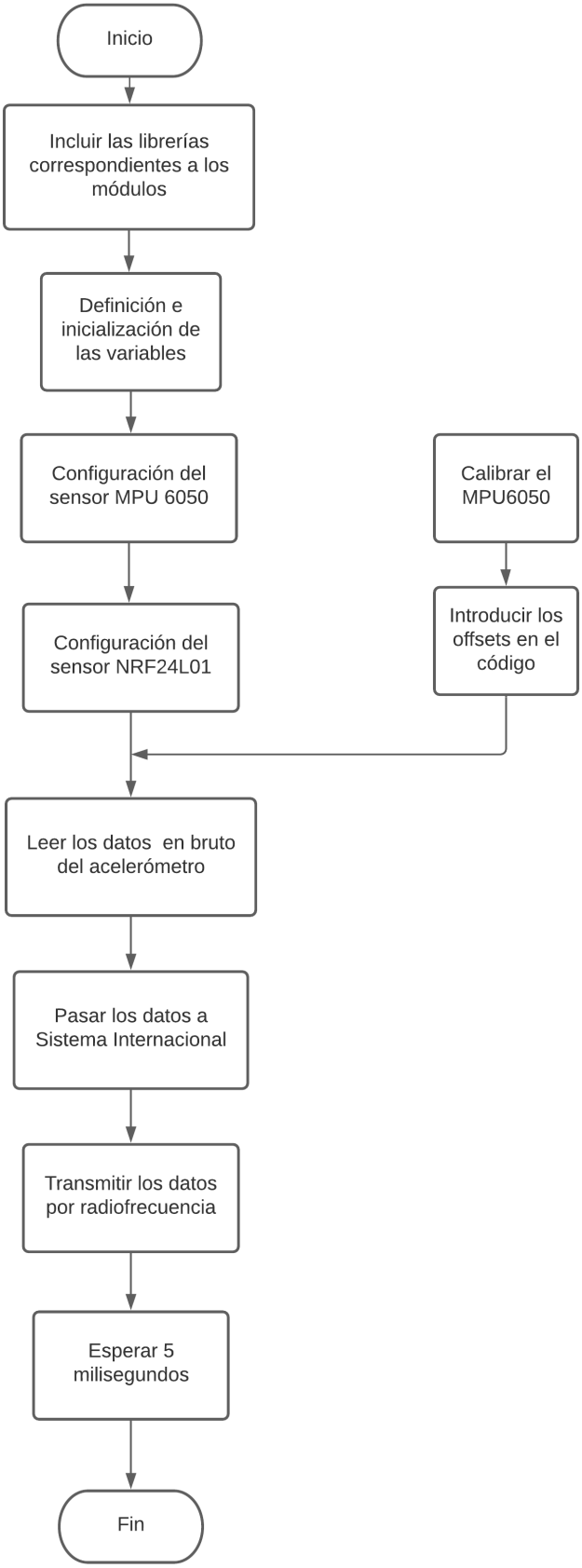


Figura 4.4 Diagrama de flujo de la placa transmisora.

4.2.3 Código del receptor

Este caso es más sencillo que el anterior, ya que la placa receptora solo se encuentra conectada al módulo de radiofrecuencia y su única función es recibir los datos y mostrarlos por puerto serie. Puede verse el diagrama de flujo correspondiente en la figura 4.5

Al tener conectado un único módulo a la placa, solo son necesarias las librerías que aseguren un comportamiento correcto del NRF24L01. Una vez hecho esto y tras configurar los pines del módulo, se comprueba si el dispositivio transmisor está disponible y, en caso de que la respuesta sea afirmativa, comienza a recibir los datos.

Es importante tener en cuenta que el valor de la variable "pipe" debe ser mismo en el código de transmisión y en el código de recepción de los datos, ya que esta variable indica el canal por el que se comunican los módulos entre sí.

Para más detalles del código, este se puede encontrar en el apéndice A.3.

4.2.4 Código de postprocesado

Este código corresponde a MATLAB, y es el responsable de procesar los datos una vez han sido tomados.

Tal y como vemos en la figura 4.6, el programa recibe los datos tomados con el Arduino en un archivo .dat, y los datos tomados con el acelerómetro de piezoeléctrico en un archivo .mat, que contiene las variables de aceleración y tiempo. El tiempo correspondiente a los datos del sistema de bajo coste se establece utilizando el periodo de muestreo utilizado en el ensayo. Posteriormente, el código filtra los datos utilizando un filtro de Chebyshev de tipo uno para reducir el valor del ruido de baja frecuencia. Además, se utiliza la FFT para representar también los datos de los ensayos pero en el dominio de la frecuencia, en lugar de en el tiempo. Finalmente, se representan los datos en ambos dominios gráficamente.

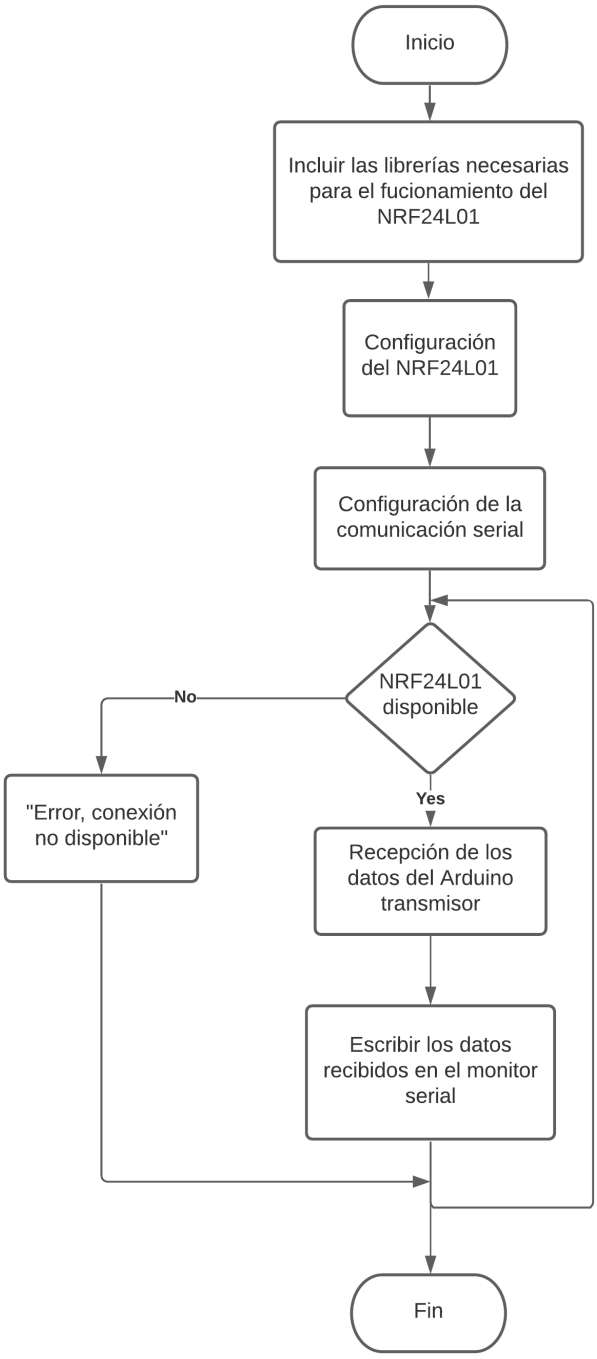


Figura 4.5 Diagrama de flujo de la placa receptora.

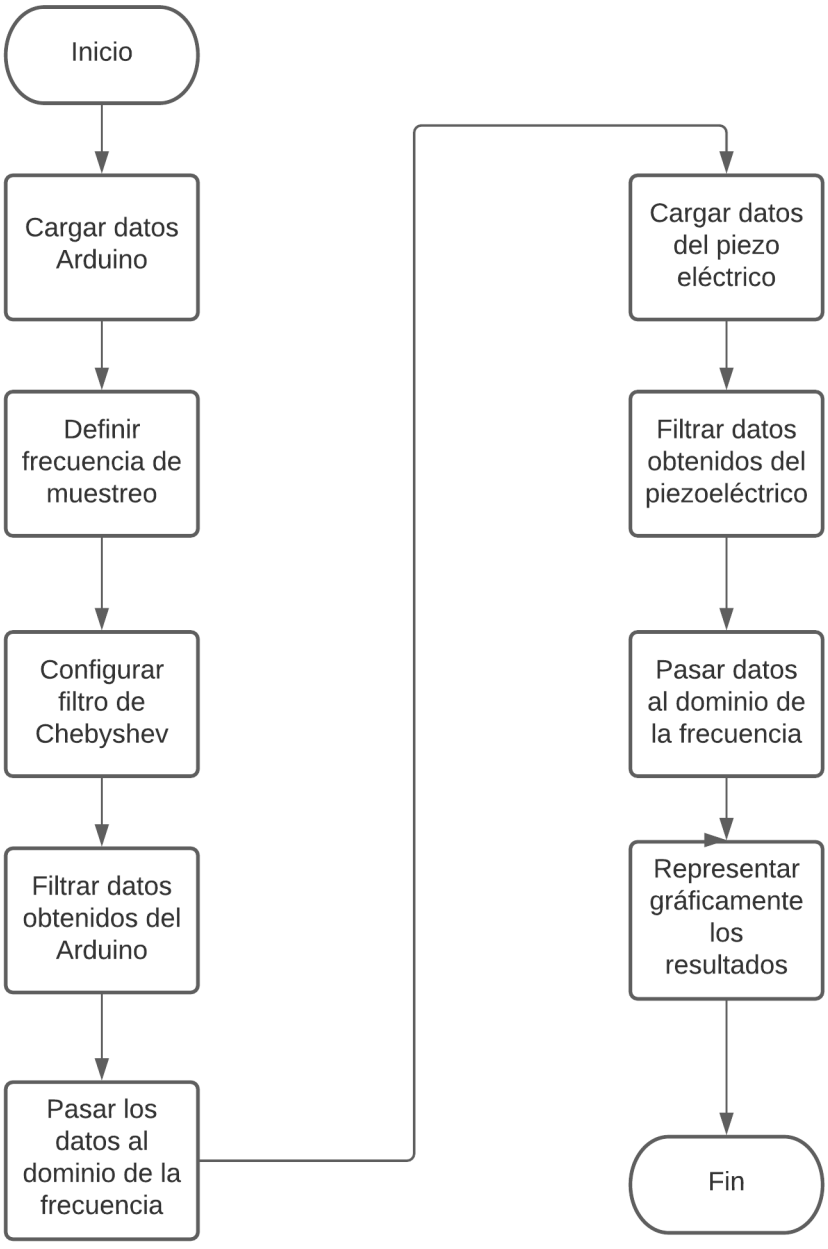


Figura 4.6 Diagrama de flujo del procesado de datos.

5 Ensayos realizados

5.1 Montaje del sistema

Lo primero que hay que realizar antes de comenzar con los ensayos es la calibración del sistema. Para ello, el Arduino transmisor debe estar conectado al ordenador mediante USB. El código de calibración se puede consultar en el anexo A.1 y su diagrama de flujo corresponde a la figura 4.3. Una vez el código haya acabado, lo cual se indicará en el monitor serial, se copian los últimos valores de Offsets que aparecen en su parte correspondiente en el código de transmisión.

Cuando la calibración se ha completado, es necesario cargar en la placa transmisora su código (Anexo A.2). Como se ha comentado anteriormente, tras valorar distintas opciones se optó por utilizar un transformador como fuente de alimentación para la placa que no queda conectada al ordenador. Para que el Arduino trabaje en remoto, basta con cargar el código desde el ordenador y, al conectarla a la corriente, utilizará automáticamente el último código que tenga guardado en su memoria. Es importante que, aunque el código comience al estar conectado, la placa de Arduino cuenta con un botón de Reset, por si fuera necesario indicar que vuelva a empezar de forma manual. El montaje final de esta parte del sistema, a falta de conectarle la alimentación, puede verse en la figura 5.1.

Una vez hecho esto, se conecta el Arduino receptor al ordenador con el cable USB y, en caso de que no esté cargado, se carga el código correspondiente (Anexo A.3). Se abre Putty y se configura tal y como aparece en las figuras 3.7. El montaje final de la placa receptora queda tal y como se ve en la imagen 5.2.

5.2 Sistemas de referencia

A modo de referencia, para comparar los resultados obtenidos con el sistema de bajo coste, se han utilizado los dispositivos disponibles en el Laboratorio de Teoría de Estructuras de la Escuela de Ingenieros de Sevilla.

5.2.1 Acelerómetro de piezoeléctrico

El sensor elegido es el acelerómetro de piezoeléctrico modelo 352C33 de la marca PCB Piezotronics. Es un acelerómetro de un único eje de medida, que en este caso corresponde al eje Z. Algunas de sus características técnicas son [23]:

- Sensibilidad: $10.2 mV/(m/s^2)$
- Rango de medida: $\pm$ 490 m/s^2

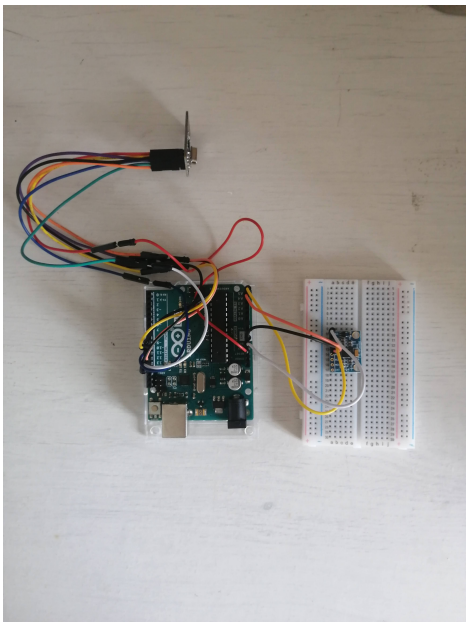


Figura 5.1 Montaje del sistema de medida.

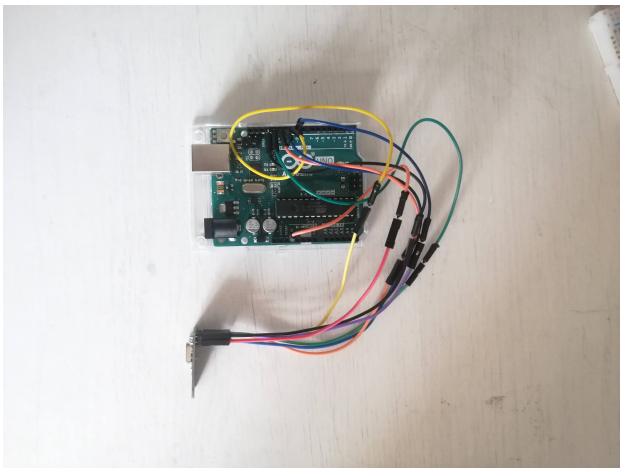


Figura 5.2 Montaje del sistema receptor.

- Rango de frecuencias: de 0.5 Hz a 10 kHz
- Frecuencia de resonancia: mayor de 50 kHz
- Voltaje de entrada: 18 a 30 V de continua
- Corriente de entrada: 2 a 20 mA
- Impedancia de salida: menor de 200 Ohmios
- Altura: 15.7mm
- Peso: 5.8 gramos
- Conector: 10-32 Jack Coaxial

Para el ensayo, el sensor se coloca en la superficie del excitador, que será la superficie vibratoria a medir. Además, se encuentra conectado al ordenador, de forma que se pueden ver los resultados obtenidos utilizando el software Bk Connect.

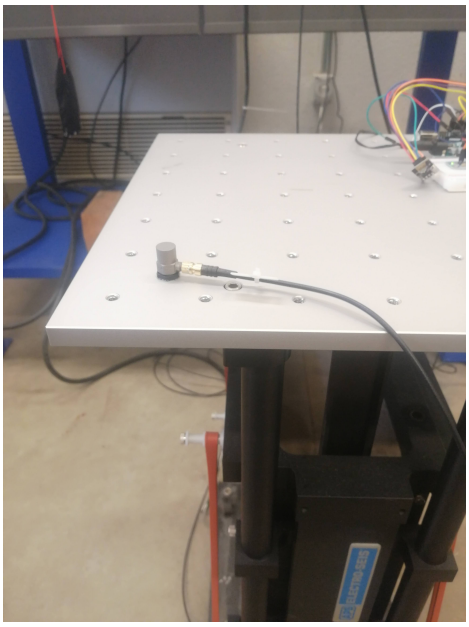


Figura 5.3 Acelerómetro de piezoeléctrico.

5.2.2 BK Connect

Este es un software de pago utilizado en este trabajo para guardar y realizar un análisis preliminar de los resultados correspondientes al acelerómetro de piezoeléctrico. Con este programa se generaron los archivos .bkc y .uff con los datos tomados en el laboratorio que, posteriormente, se guardaron en archivos .mat utilizando MATLAB para facilitar su análisis.

5.3 Ensayos

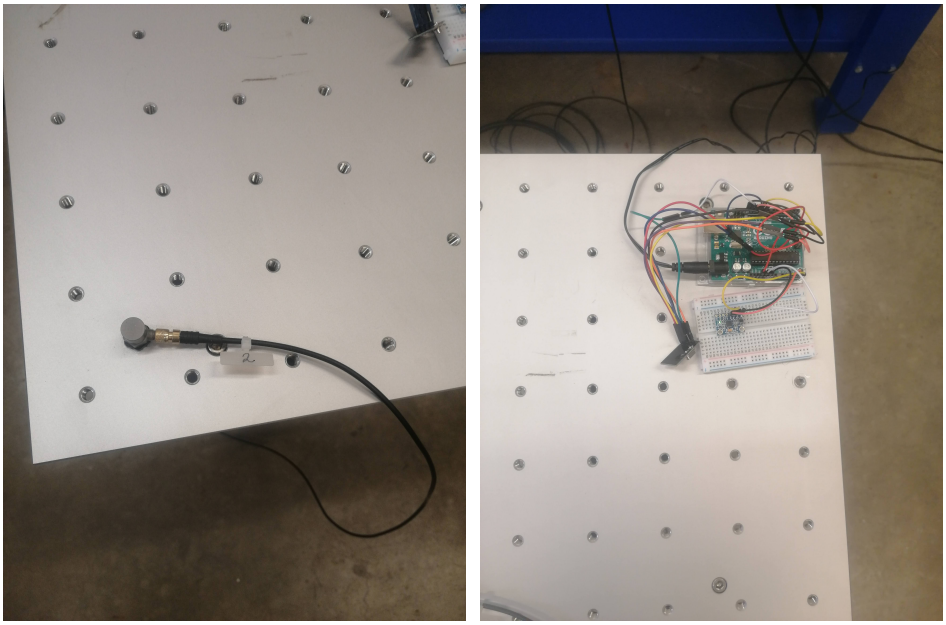
Para realizar los ensayos se montaron tanto el acelerómetro de piezoeléctrico como el sistema formado por el Arduino transmisor y el sensor MPU-6050 sobre la superficie de un excitador de la marca APS Dynamics.

Los ensayos se realizaron con vibraciones de diferentes características. Para ello, se tomaron valores de ambos acelerómetros durante ensayos con distintas frecuencias y distintas amplitudes de vibración. Finalmente, se realizaron dos Burst Random con dos amplitudes diferentes. Los resultados de estas vibraciones se guardaron en sus archivos correspondientes y, posteriormente, se procesaron mediante el código descrito en el anexo B.

5.4 Resultados obtenidos

Para realizar los ensayos se utilizó como señal de excitación señales senoidales de distinta frecuencia y amplitud. Respecto a las frecuencias, se utilizaron 10, 20, 30, 40 y 50 Hz. En el caso de las amplitudes, se utilizó 100mV y 200mV. Finalmente, se realizaron dos ensayos utilizando un Burst Random, con las amplitudes ya mencionadas.

En las siguientes secciones, se presentan las gráficas correspondientes a los ensayos realizados. Se presentan tanto los datos obtenidos con el sistema de bajo coste, como los datos obtenidos con el acelerómetro de piezoeléctrico. Además, para ambos casos se incluye la gráfica en el dominio de la frecuencia, a modo de comparación.



(a) (b)

Figura 5.4 Montaje de los acelerómetros sobre el excitador.

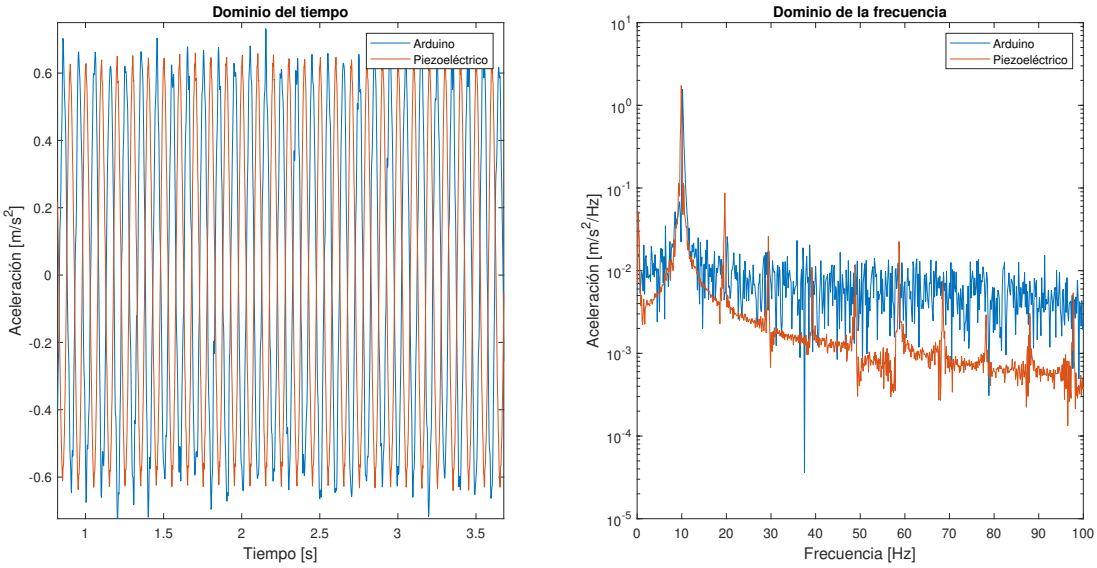


Figura 5.5 Ensayo con 10 Hz de frecuencia y 100 mV de amplitud.

En el caso del dominio temporal, puede verse en las gráficas que, a grandes rasgos los resultados son similares entre ambos sistemas. El orden de magnitud de las medidas del Arduino es correcto, sin embargo, los resultados obtenidos en ciertos puntos sí difieren unos de otros. En el dominio de la frecuencia queda representado que el sistema de bajo coste detecta la componente principal de la vibración en cada ensayo, pero falla al detectar las componentes secundarias del mismo, que sí son detectadas por el sistema de piezoeléctrico. Además, la cantidad de ruido reibida por el sistema de Arduino es considerablemente mayor que la recibida por el sistema del laboratorio.

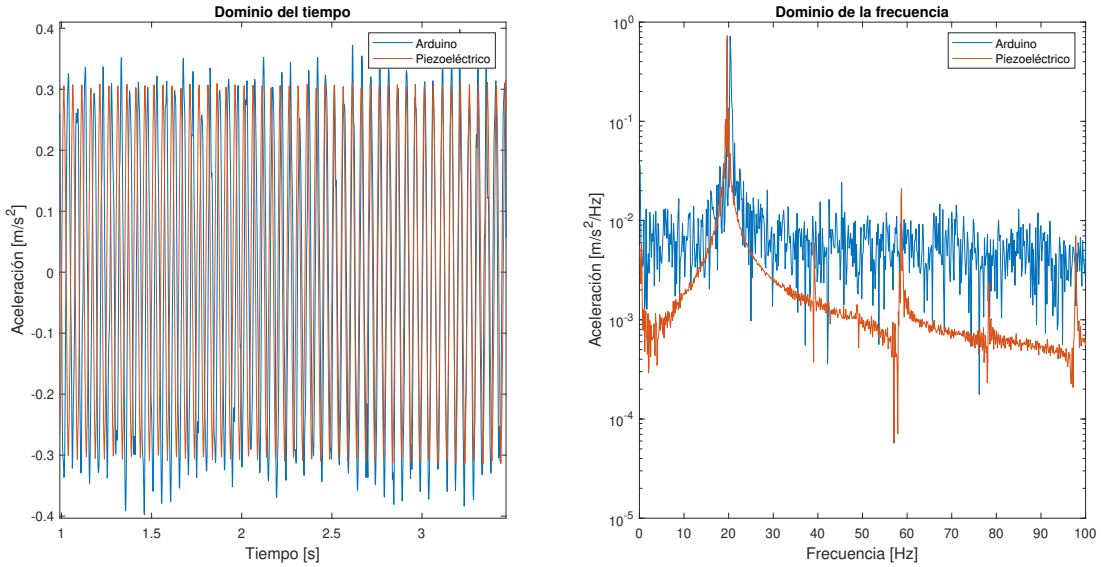


Figura 5.6 Ensayo con 20Hz de frecuencia y 100mV de amplitud.

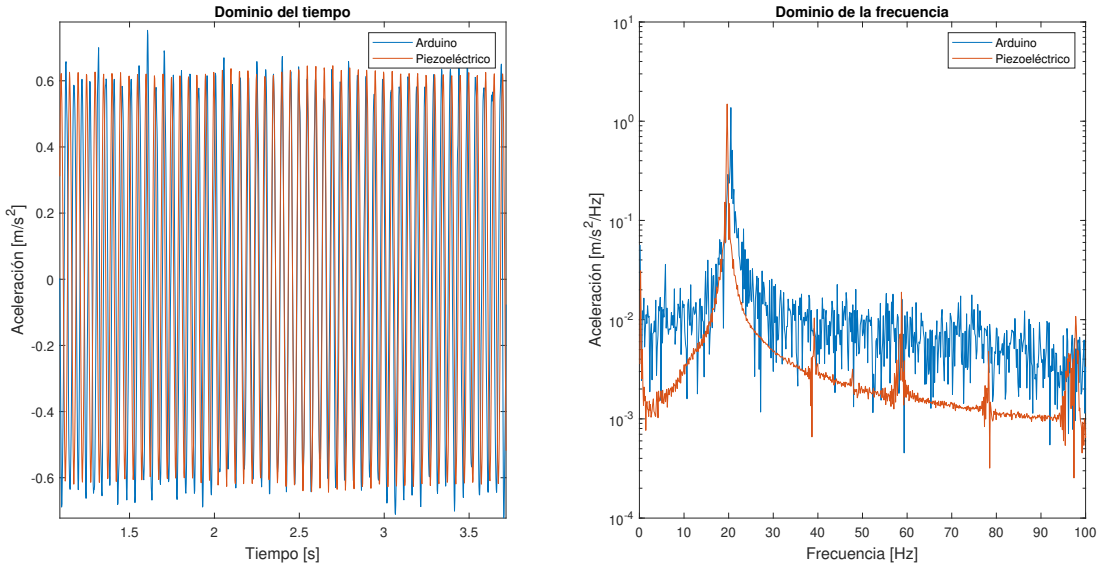


Figura 5.7 Ensayo con 20Hz de frecuencia y 200mV de amplitud.

En el caso de los dos ensayos de burst random, de nuevo, el ruido detectado por el sistema de bajo coste es considerablemente mayor que el obtenido por el acelerómetro de piezoeléctrico como puede verse en la zona en la que el sistema no está excitado. En el dominio de la frecuencia, sin embargo, puede verse una gran similitud entre ambos espectros hasta una frecuencia de 50Hz. A partir de ese punto, ambas gráficas difieren considerablemente.

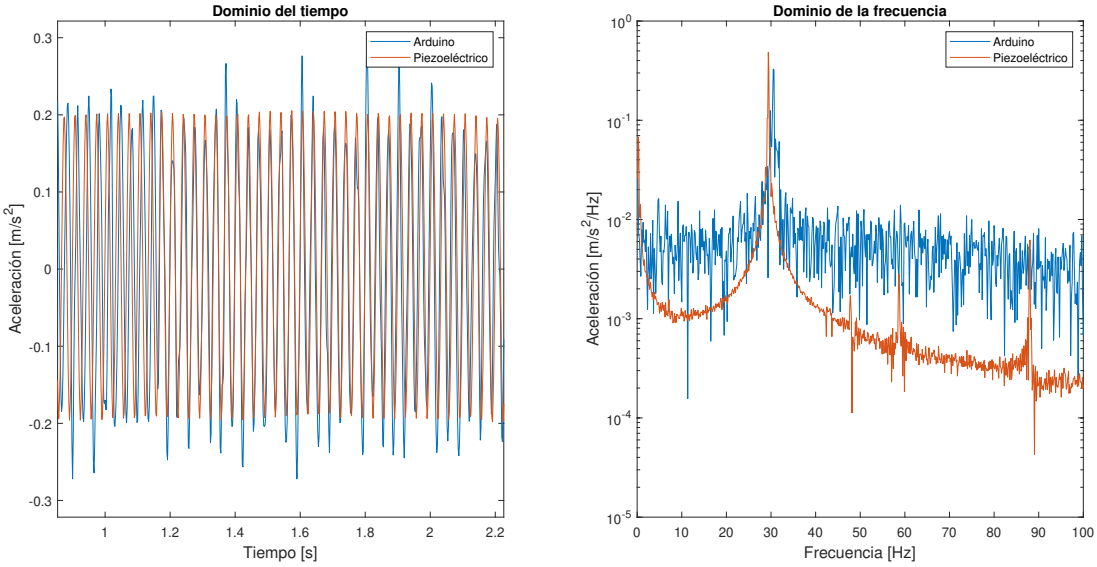


Figura 5.8 Ensayo con 30Hz de frecuencia y 100mV de amplitud.

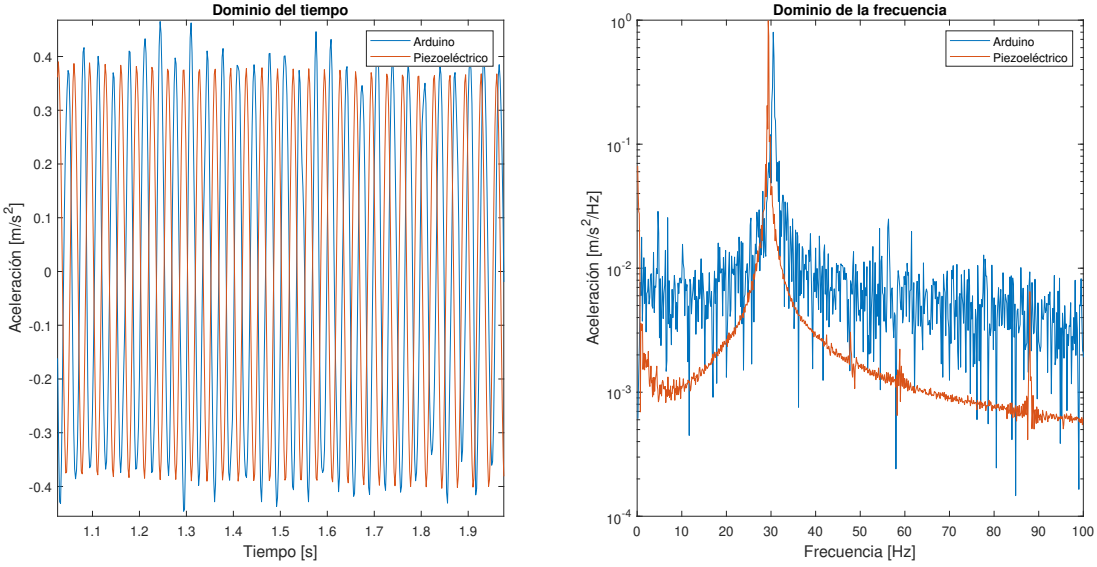


Figura 5.9 Ensayo con 30Hz de frecuencia y 200mV de amplitud.

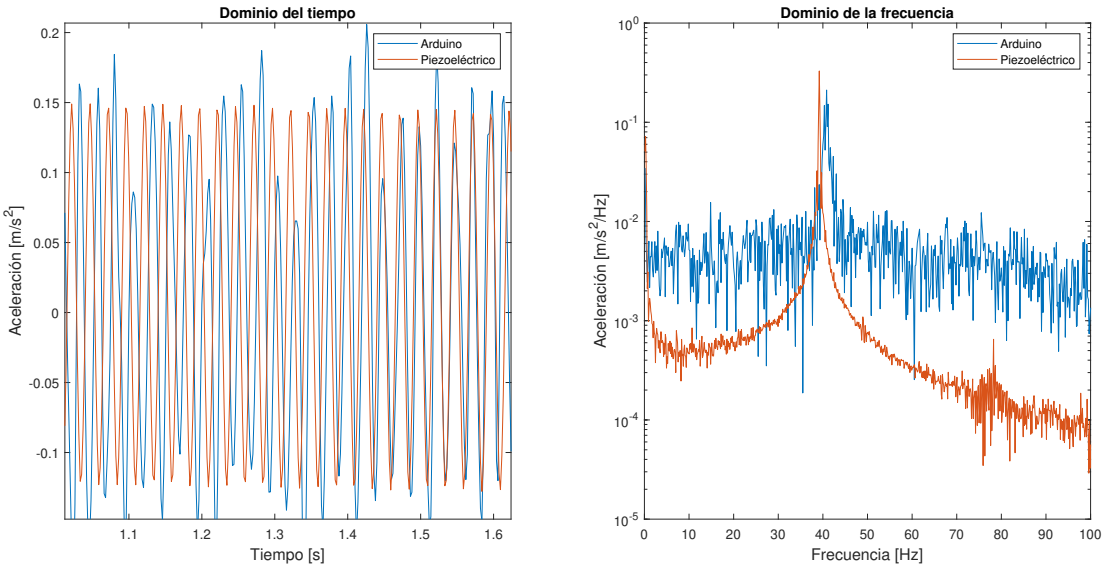


Figura 5.10 Ensayo con 40Hz de frecuencia y 100mV de amplitud.

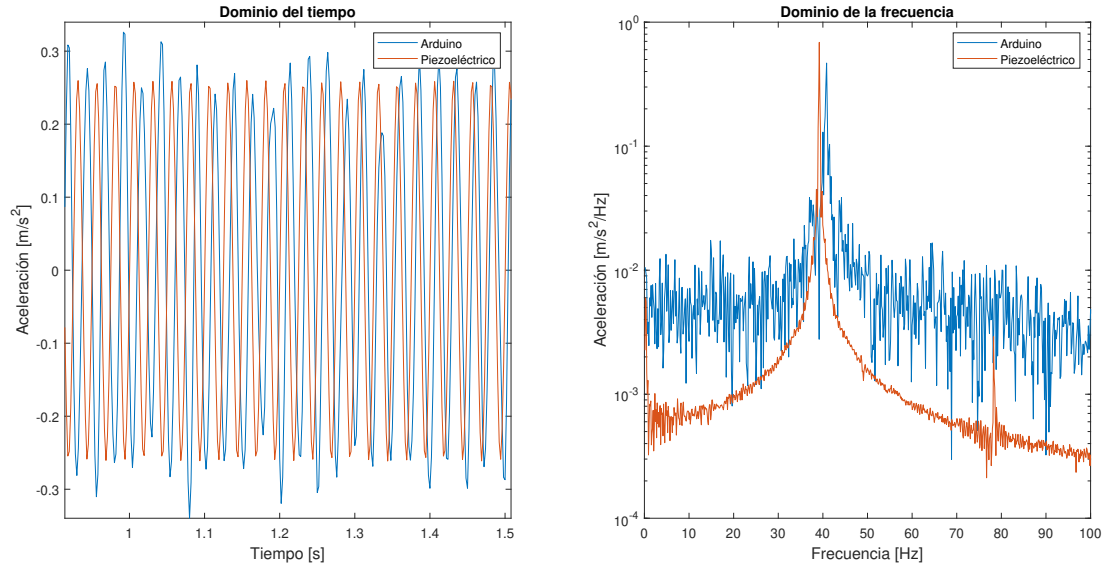


Figura 5.11 Ensayo con 40Hz de frecuencia y 200mV de amplitud.

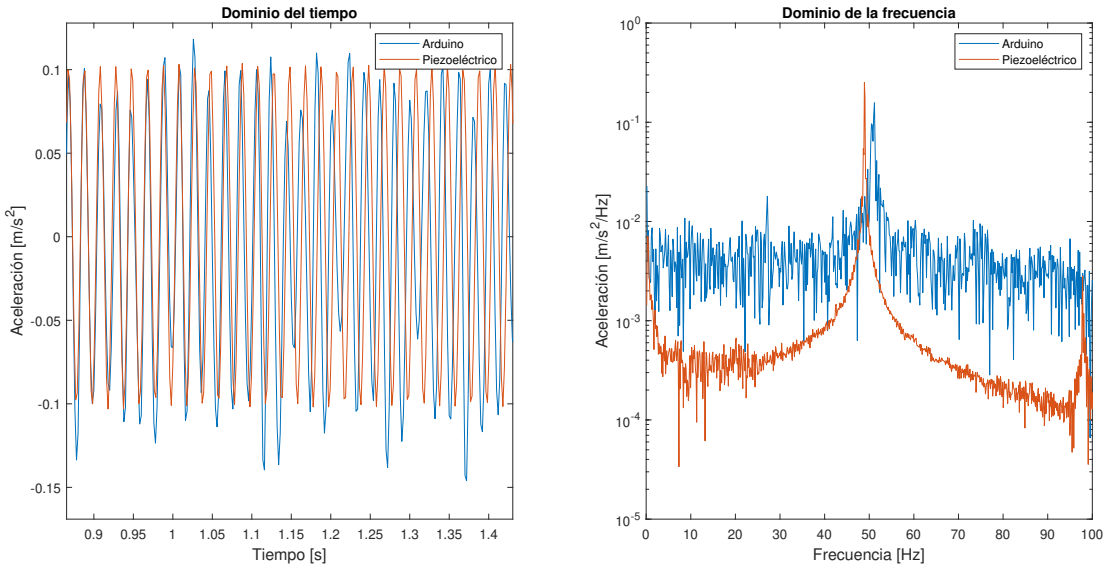


Figura 5.12 Ensayo con 50Hz de frecuencia y 100mV de amplitud.

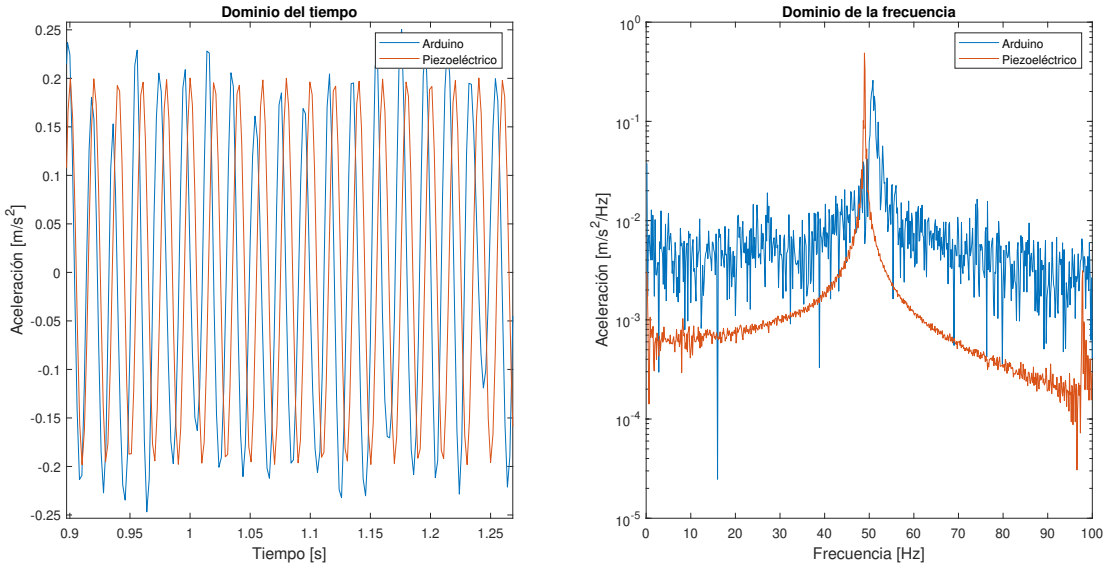


Figura 5.13 Ensayo con 50Hz de frecuencia y 200mV de amplitud.

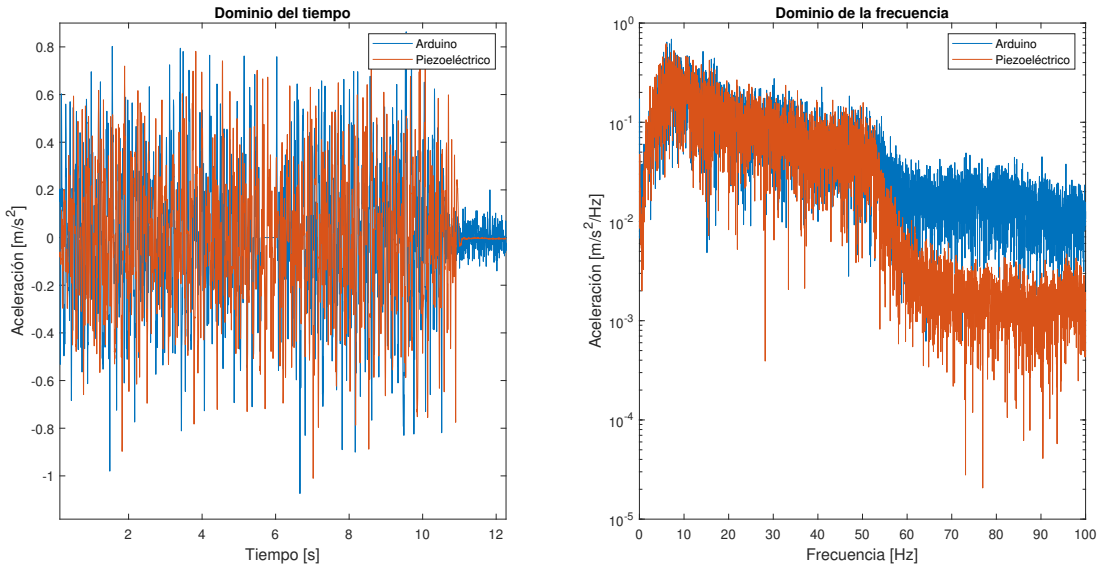


Figura 5.14 Ensayo con excitación Burst random y 100mV de amplitud.

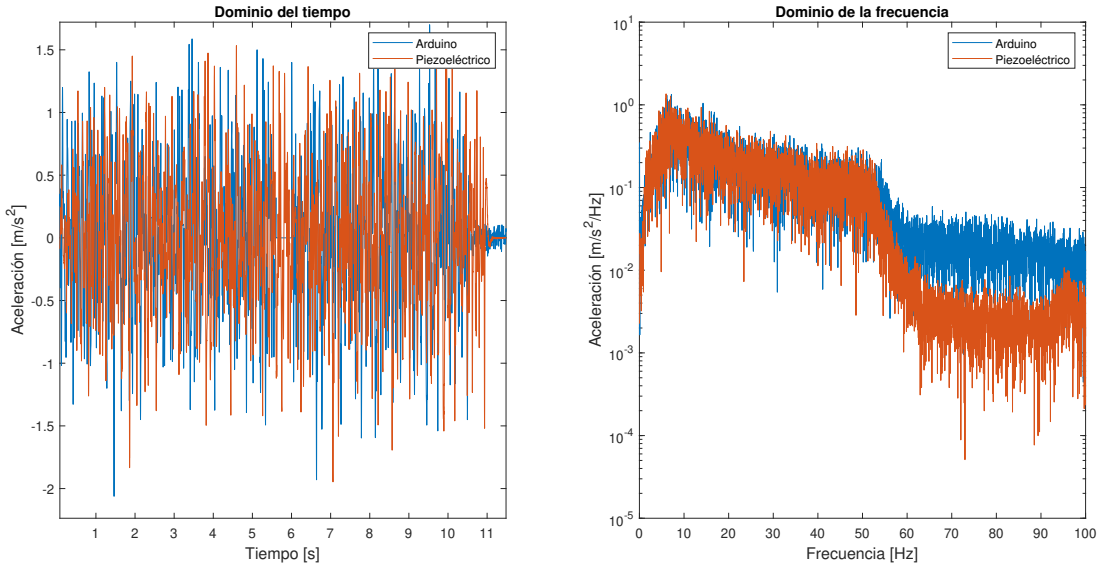


Figura 5.15 Ensayo con excitación Burst random y 200mV de amplitud.

6 Conclusiones

Los resultados obtenidos en este trabajo demuestran que el desarrollo de un sistema de bajo coste para medir vibraciones es posible y realizable. Ahora bien, es necesario tener en cuenta para ello que los sistemas tienen limitaciones importantes. Una de las grandes limitaciones es que el sensor aquí utilizado tiene una precisión inferior al sistema de piezoeléctrico. Si esta reducción de la calidad de las medidas frente al precio del mismo es aceptable o no, dependerá en general de la aplicación que vaya a hacerse del mismo. Si el sistema se desarrolla para realizar análisis a grandes rasgos de una estructura, sí es un sistema útil que cumple su función. En el caso de que el sistema pretenda utilizarse en aplicaciones que necesiten un grado muy fino de ajuste, existirían riesgos que habría que valorar.

Dentro de las limitaciones comentadas, sin embargo, algunas tienen solución sencilla. Una de las limitaciones es que, al enviar los datos por radiofrecuencia en tiempo real existe un rango máximo de distancia en el que esto es posible, lo cual dependerá del alcance de los transceptores. Por tanto, siempre habrá que mantener el Arduino receptor dentro de este rango, o se perderá la conexión. Una solución a esto sería sustituir el envío de datos por el guardado de los mismos en una memoria utilizando un módulo para tarjetas SD o microSD, o realizar ambas funciones simultáneamente de forma que, en caso de que no se encuentre dentro del rango de los módulos de radiofrecuencia, los datos queden guardados en la tarjeta SD a modo de respaldo. Otra solución sería utilizar módulos con un alcance mayor, como es el caso de la otra versión disponible de los módulos NRF24L01 aquí utilizados, o sustituirlos por otras tecnologías.

Aun así, no todo son desventajas. El pequeño tamaño de los dispositivos de bajo coste utilizados en este trabajo le confieren una gran capacidad de transporte y portabilidad.

En este caso se optó por una fuente de alimentación, ya que los ensayos se han probado dentro del laboratorio. Sin embargo, es importante tener en cuenta que si el ensayo se va a realizar en una estructura real que se encuentre en el exterior, será necesario utilizar una batería u otro sistema de alimentación autónomo. Además, en caso de que se busque una monitorización a largo plazo, el sistema de alimentación debe asegurar el funcionamiento durante todo el proceso.

6.1 Posibles desarrollos futuros

En este trabajo se optó por el módulo MPU-6050, que es muy económico, sin embargo sería interesante realizar un estudio con acelerómetros de mejor calidad, tratando de ver si los resultados son más exactos. Otra opción sería comparar los resultados aquí presentados con los resultados obtenidos utilizando un sensor de vibraciones de bajo coste (como por ejemplo, el módulo SW420 o el módulo 801S).

Como se ha comentado anteriormente, sería una buena línea de trabajo incluir en el montaje un módulo para tarjetas de memoria, o bien como respaldo o bien como sistemas principal de guardado de datos.

En este trabajo se ha utilizado MATLAB para realizar el procesado de los datos obtenidos en los ensayos. Esto ha sido posible debido a que la Universidad dispone de licencias de dicho programa. Sin embargo, si no se dispusiera de estas licencias se complicaría el desarrollo del mismo. Por tanto, sería interesante realizar una aplicación que permita el análisis de los datos sin utilizar un software que necesite licencia.

Además, también existe la posibilidad de realizar desde cero la placa de circuitos integrados. De esta forma, utilizando un software de diseño de placas (por ejemplo, Eagle) y soldando los componentes necesarios a ella, se podría reducir el espacio necesario para el sistema. Entre las ventajas de esto se encuentra que se podría prescindir de la placa de pruebas o protoboard, no serían necesarios los cables que conectan los pines y, por tanto, se elimina la posibilidad de un fallo por una mala conexión o un mal funcionamiento de los mismos.

Apéndice A
Códigos de Arduino

A.1 Código de calibración

```
0       // Librerías
  #include "I2Cdev.h"
2 #include "MPU6050.h"
  #include "Wire.h"
4
  const int dir = 0x68;
6       // Si el pin A0 está a 1, dir = 0x69
  MPU6050 sensor(dir);
8
        // Se crean las variables de las aceleraciones en los ejes
10 int ax, ay, az;
  int ax_o, ay_o, az_o;
12
  void setup() {
14      Serial.begin(57600);
        Wire.begin();
16      sensor.initialize();
        if (sensor.testConnection()){
18          Serial.println("El sensor se ha iniciado correctamente, va a comenzar la calibración.");
            Serial.println("Por favor, mantenga el sensor horizontal y no lo mueva");
20            }
        else{
22          Serial.println("Inicializaci ón fallida. Por favor, vuelta a intentarlo");
              }
24      sensor.getAcceleration(&ax,&ay,&az);
        ax_o = sensor.getXAccelOffset();
26      ay_o = sensor.getYAccelOffset();
        az_o = sensor.getZAccelOffset();
28      }

30      void loop() {
      sensor.getAcceleration(&ax,&ay,&az);
32  delay(30000);
      if ((abs(ax<0.5)) && (abs(ay<0.5)) && (abs(az<0.5))) {
34      Serial.println("Los Offset necesarios son:");
        Serial.println(ax_o);
36      Serial.println(ay_o);
        Serial.println(az_o);
38      delay(12000000);
      }
```

```
40    else {
        Serial.print("hola");
42      //Bucle para el Offset en el eje X
        if (ax>0.02) {
44        ax_o--;
          Serial.println(ax_o);
46      }
        else if (ax<0.02){
48        ax_o++;
          Serial.println(ax_o);
50      }
        sensor.setXAccelOffset(ax_o);
52
        //Bucle para el Offset en el eje Y
54      if (ay>0.02) {
          ay_o--;
56      }
        else if (ay<0.02){
58        ay_o++;
        }
60      sensor.setYAccelOffset(ay_o);

62      //Bucle para el Offset en el eje Z
        if (az>0.02) {
64        az_o--;
        }
66      else if (az<0.02){
          az_o++;
68      }
        sensor.setZAccelOffset(az_o);
70    }
  }
```

Código A.1 Calibración del sensor MPU6050.

A.2 Código transmisor

```
0
  //Librerías necesarias para el funcionamiento del MPU6050
2 #include "I2Cdev.h"
  #include "MPU6050.h"
4 #include "Wire.h"

6 //Librerías necesarias para el funcionamiento del NRF24L01
  #include "nRF24L01.h"
8 #include "RF24.h"
  #include "RF24_config.h"
10 #include "SPI.h"


12

14 //Configuración de NRF24L01
  int CE = 9; //Pin CE del módulo
16 int SCN = 10; //Pin CSN del módulo
  RF24 radio(CE,SCN);
18 const uint64_t pipe = 0xE8E8F0F0E1LL;

20 //Configuración del sensor MPU6050
```

```
const int dir = 0x68;
// En caso de colocar el pin A0 a 1, dir = 0x69
MPU6050 sensor(dir);

// Creación de las variables necesarias
float aceleracion[2];
int ax,ay,az;
float az_si;
//


// Es necesario usar factores de conversión para obtener las medidas en sistema internacional

const float factor_accel = 2*9.81/32768;
// Si lo queremos en g, 1g=9.81m/s2

// Se define el periodo necesario para la frecuencia de muestreo deseada

// En este caso la frecuencia de muestreo es 500Hz
int periodo = 2; // En millis
unsigned long tiempo = 0;


void setup(void) {
  Serial.begin(115200);
  Wire.begin();
  radio.begin();
  radio.openWritingPipe(pipe); // Comienza a transmitir datos

  sensor.initialize();
  if (sensor.testConnection()){
    Serial.println("El sensor se ha iniciado correctamente");
  }
  else {
    Serial.println("Inicialización fallida. Por favor, vuelta a intentarlo");
  }

  // Introducir aquí los resultados obtenidos tras calibrar el sensor
  sensor.setXAccelOffset(-4790);
  sensor.setYAccelOffset(979);
  sensor.setZAccelOffset(-297);
}


void loop(void) {

    tiempo = millis();

    sensor.getAcceleration(&ax, &ay, &az); // Se obtienen los datos del MPU6050

    // Se convierten a Ssistema Internacional
    aceleracion[1]= factor_accel * az;

    radio.write(&aceleracion, sizeof aceleracion);
    // Se envían los datos utilizando el módulo de radiofrecuencia

    while (millis() < tiempo + periodo) {}
    }
```

Código A.2 Código transmisor.

A.3 Código receptor

```
0
  // Librerías necesarias para el funcionamiento del NRF24L01
2 #include "nRF24L01.h"
  #include "RF24.h"
4 //#include "RF24_config.h"
  #include "SPI.h"
6
  // Configuración del módulo NRF24L01
8 int CE = 9; // Pin CE del módulo
  int SCN = 10; // Pin SCN del módulo
10 RF24 radio(CE,SCN);
  const uint64_t pipe = 0xE8E8F0F0E1LL;
12
  // Se define la variable
14 float aceleracion [2];


16
  void setup(void) {
18   radio.begin();
     Serial.begin(115200);
20   radio.openReadingPipe(1,pipe);
     radio. startListening (); // Comienza a recibir
22 }

24 void loop(void) {
       if (radio. available ()){
26       radio.read(&aceleracion, sizeof aceleracion);
         Serial. println ( aceleracion [1]);
28
         }
30


32   }
       else{
34     Serial. println ("Error, no se ha podido establecer conexión con el transmisor");
     }
```

Código A.3 Código receptor.

Apéndice B
Código de MATLAB

```
clc
clear all
close all

load('C:\Users\Paloomm\Documents\Universidad\tfg\Valores\Arduino20_100.mat'); %Cargar el archivo de datos del arduino


fs=500;                 %frecuencia de muestreo (número de datos por segundo registrados por el sistema) [Hz]
deltat =1/fs;           %intervalo (período) de tiempo [s] en el que se registran los datos
time= deltat : deltat : size(datos,1)*deltat;


%Filtro de paso alto para reducir el ruido de baja frecuencia
    ChebyOrder2=3;
    ChebyRipple2=0.5;
    ChebyFreq2=19.8; %Frecuencia de corte del filtro
    [Bf2,Af2]=cheby1(ChebyOrder2,ChebyRipple2,ChebyFreq2/fs*2,'high');

%Se filtran los resultados del Arduino
    resp1= filter (Bf2,Af2,datos,[],1) ;

%Dominio de la frecuencia
     data_fft = fft(resp1);
    nfreq=size( data_fft ,2);
    df=fs/nfreq;
     freq=df:df:fs;

%Se cargan los datos del piezoeléctrico, guardados en un archivo .mat
%El archivo contiene las valiables aceleración y tiempo
load('C:\Users\Paloomm\Documents\Universidad\tfg\Paloma\resultados_seno_20Hz100mV.mat');

%Se filtran los resultados del piezoeléctrico para reducir el ruido de baja frecuencia
resp2= filter (Bf2,Af2, aceleracion ,[],1) ;

%Dominio de la frecuencia
     data_fft2 = fft(resp2);
    nfreq2=size( data_fft2 ,2);
    df2=fs/nfreq2;
    freq2=df2:df2:fs;


```

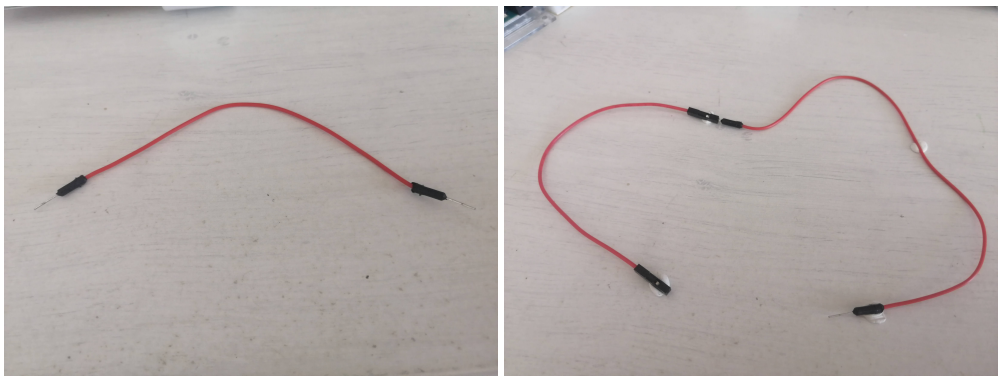
```
%Representación gráfica

%Se representarán los resultados en dos gráficas, una para el dominio del
%tiempo y otra para el dominio de la frecuencia
 tiledlayout (1,2)

%Representación en el dominio del tiempo
 nexttile
     plot(time, resp1)    %Representación de los datos del Arduino
     hold on
     plot(tiempo,resp2)   %Representación de los datos del piezoeléctrico
     ylim([-max(abs(resp1))*1.1 max(abs(resp1))*1.1])
     ylabel('Aceleración [m/s^2]','Fontsize',12);
     xlabel('Tiempo [s]','Fontsize',12);
      title ('Dominio del tiempo');
     legend('Arduino','Piezoeléctrico');

%Representación en el dominio de la frecuencia
 nexttile
     plot(freq(1:(nfreq)/2),abs(data_fft(1:(nfreq)/2))/fs); %Representación de los datos del Arduino
     hold on
     plot(freq2(1:(nfreq2)/2),abs(data_fft2(1:(nfreq2)/2))/fs)%Representación de los datos del piezoel
     éctrico
     xlabel('Frecuencia [Hz]','FontSize',12);
     ylabel('Aceleración [m/s^2/Hz]','FontSize',12)
     set(gca,'yscale','log') %Escala logarítmica
     xlim([0 100])
     legend('Arduino','Piezoeléctrico');
      title ('Dominio de la frecuencia');

%Por último, salen por pantalla los resultados máximos de cada dispositivo
     fprintf('\nMáximo registro ard [m/s^2]: %f\n\n',max(abs(resp1)))
     fprintf('\nMáximo registro piezoeléctrico [m/s^2]: %f\n\n',max(abs(resp2)))
```

Código B.1 Procesado de datos.

Apéndice C
Montaje detallado

En este anexo se va a tratar de realizar una descripción más detallada del montaje del sistema paso a paso.

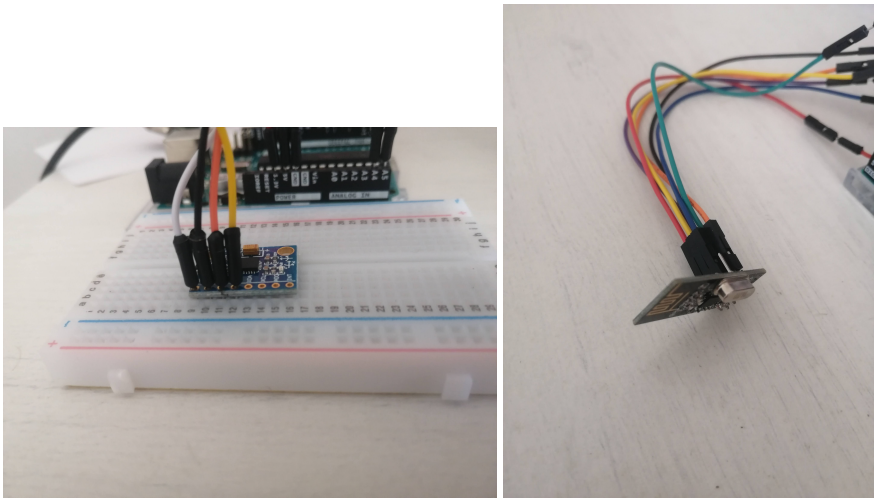
Cada Arduino viene en una caja de cartón, en la que se incluye únicamente la placa y una breve documentación. Para conectar la placa al ordenador, se utiliza un cable USB A/B que es necesario comprar por separado. Para conectar los componentes a la placa se utilizan cables como los que pueden verse en C.1a y C.1b.



(a) Cable macho/macho. **(b)** Cable macho/hembra.

Figura C.1 Cables de conexión para Arduino.

El primer paso es conectar las placas Arduino con sus módulos correspondientes. En el extremo de Arduino, los cables macho se insertan en el pin corresondiente. La conexión con el módulo MPU-6050 se realiza utilizando la placa de pruebas y los cables macho/macho. En la figura C.2a puede verse el resultado final del módulo con los cables conectados a la placa de pruebas. Por su parte, el módulo NRF24L01 cuenta con pines ya soldados, por tanto, lo más sencillo es conectarlo directamente al Arduino utilizando cables macho/hembra, de forma que el extremo macho queda conectado al Arduino, y el extremo hembra queda conectado al módulo, como representa la figura C.2b. De esta forma, la placa que funcionará como transmisora quedará conectada a un módulo NRF24L01 y al sensor MPU6050; mientras que la que funcionará como receptora quedará conectada únicamente a un módulo NRF24L01. Las conexiones se detallan en 4.1 y 4.2 y pueden verse esquemáticamente en las figuras C.3 y C.4.



(a) Módulo MPU-6050 con los cables conectados. **(b)** Módulo NRF24L01 con los cables conectados.

Figura C.2

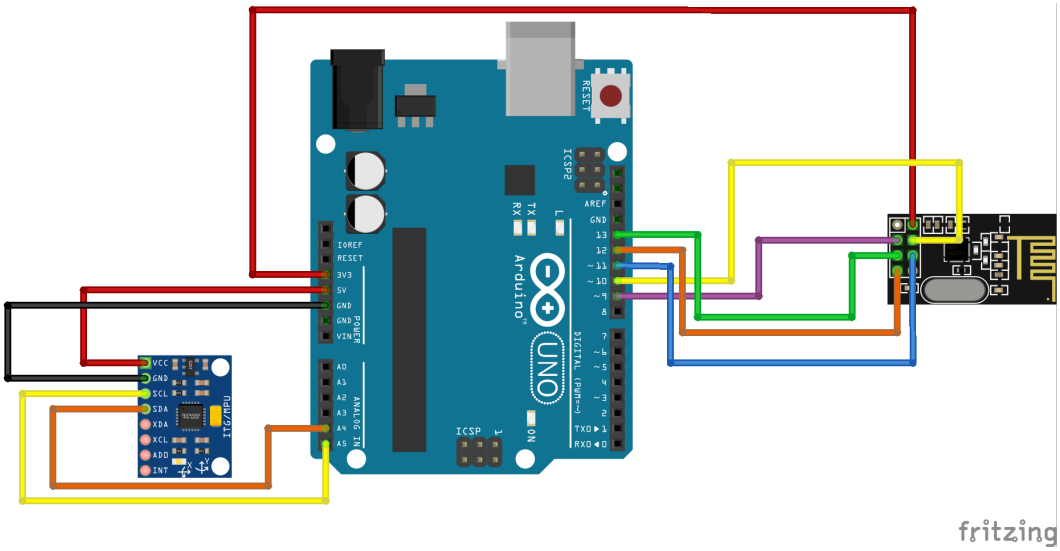


Figura C.3 Esquema de conexión del transmisor.

Una vez conectados los dispositivos, lo siguiente es colocar la placa transmisora sobre la superficie de la cual se quieren medir las aceleraciones, como se ve en la figura 5.4b colocada sobre el excitador. Se conecta la placa transmisora al ordenador y, manteniendo el sistema quieto y sin cambiarlo de posición, se carga en ella el código de calibración que se detalla en el anexo A.1. Para cargarlo, basta con abrir el código en IDE, comprobar que el puerto seleccionado es el puerto en el que está conectada la placa y pulsar sobre el icono de flecha horizontal que se encuentra en la parte superior izquierda. El puerto seleccionado aparece indicado en la esquina inferior izquierda de IDE.Para cambiarlo, simplemente se selecciona Herramientas en el menú desplegable superior, y aparece la opción Puerto, como puede verse en la imagen C.5. El puerto en el que esté conectada la placa se llamará "Arduino UNO".

Para ver los resultados de la calibración, que se envían por puerto serie, se abre el monitor serial. Para ello, de nuevo, se pulsa en la pestaña desplegable "Herramientas" y se selecciona la opción

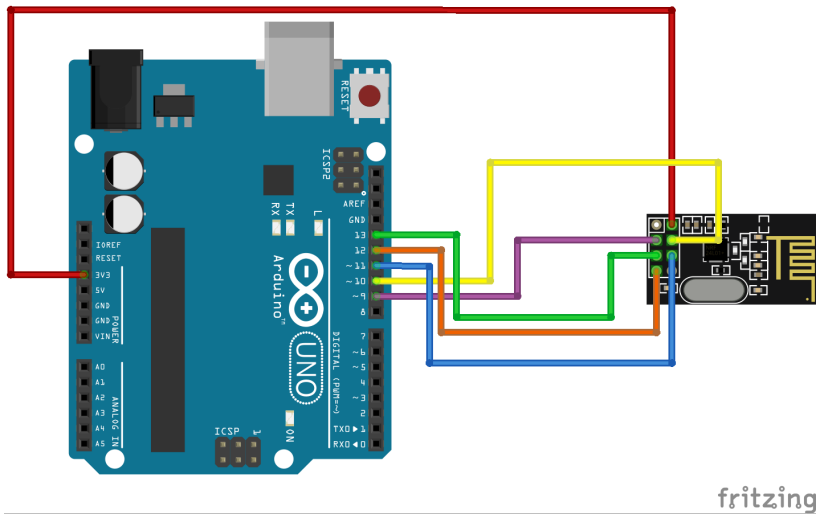


Figura C.4 Esquema de conexión del receptor.

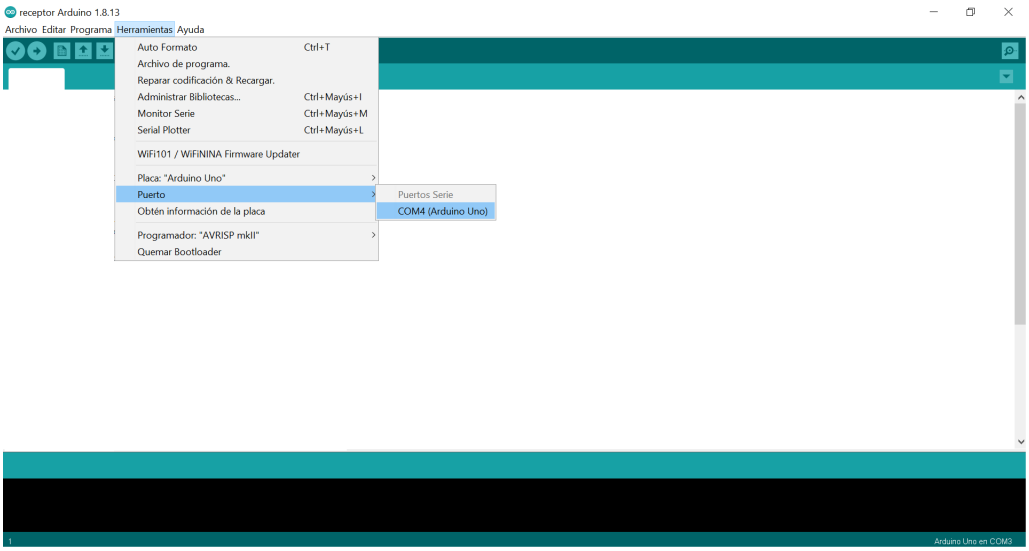


Figura C.5 Cambiar Puerto Serie.

Monitor Serie. Una vez abierto, comenzarán a aparecer por pantalla los resultados. Cuando el código termine lo avisará por pantalla, mostrando además los offsets que será necesario copiar en el código de transmisión (Anexo A.2), en las líneas que se indican en dicho código. Por último, se carga el código de transmisión en la placa transmisora, con los offsets acutalizados y ya puede desconectarse del ordenador.

Se conecta posteriormente la placa receptora al ordenador con el cable USB A/B y se carga en ella el código necesario para la recepción de datos, descrito en el anexo A.3. De nuevo, es importante comprobar que el puerto seleccionado es el puerto correcto en el que la placa está conectada, ya que en caso contrario saltará un error por pantalla.

Con ambos códigos cargados en sus correspondientes placas, se conecta el Arduino transmisor a la corriente eléctrica mediante el cable que se ve en la figura 4.2b, que se conecta al puerto Jack de la placa. Cuando está conectada a la corriente, la luz de encendido de la placa, un LED de color

verde, se encuentra encendida. Si esto no ocurre, será necesario comprobar que la placa y el cable funcionan correctamente. En este paso, la placa receptora no interviene, de modo que se mantiene conectada al ordenador.

Lo siguiente es abrir Putty y configurar el programa tal y como se ve en las figuras 3.7a y 3.7b. En el campo "Serial Line" se escribe el puerto al que está conectada la placa receptora, que puede consultarse utilizando IDE como se ha explicado anteriormente. De la misma forma, en "Speed" se escribe la velocidad a la que se ha configurado la comunicación por Puerto Serie en el código, que en este caso es 115200 baudios.

En el menú izquierdo, se selecciona la opción "Logging" y se abre otro menú de opciones distinto al anterior. En "Session Logging" es necesario seleccionar la opción "All session output", para que se guarden en el archivo todos los datos recibidos. En "log file name", se indica la ruta en la que se desean guardar los datos, indicando al final de esta el nombre del archivo que se desee y, su extensión (en este caso, .dat).

Al pulsar "Open", se abre el monitor serial, que es igual que el que puede verse en 3.8. Una vez la sesión haya comenzado, se recibirán los datos que irán apareciendo por pantalla y que se guardarán en el archivo deseado. Cuando la sesión se cierre, el archivo quedará guardado en la ruta que se ha indicado en la configuración.

Una vez tomados los datos en todos los ensayos, basta con desconectar cada Arduino de su alimentación y de sus componentes. Para volver a programarlo con posterioridad no hay que realizar ninguna opción adicional, bastará con cargar el código que se desee en él.

Índice de Figuras

1.1 Tipos de giróscopos respecto a su precio [5] 2

2.1 Logo Arduino.CC 5
2.2 Placa Arduino UNO 6
2.3 Diagrama de pines de la placa Arduino UNO 7
2.4 Módulo MPU6050 8
2.5 Módulo NRF24L01 10
2.6 Placa de pruebas 11
2.7 Conexiones en una placa de pruebas 12

3.1 IDE 13
3.2 Opciones disponibles en IDE 14
3.3 Documentación sobre las funciones disponibles en la web de Arduino 14
3.4 Librerías 14
3.5 Monitor serial del IDE 15
3.6 Serial Plotter del IDE de Arduino 15
3.7 Configuración de Putty 16
3.8 Monitor serial de Putty 16
3.9 Pantalla principal de MATLAB 17

4.1 Esquema de las conexiones del sistema. 19
4.2 Posibilidades valoradas para alimentar el Arduino en remoto 21
4.3 Diagrama de flujo de la calibración 22
4.4 Diagrama de flujo de la placa transmisora 23
4.5 Diagrama de flujo de la placa receptora 25
4.6 Diagrama de flujo del procesado de datos 26

5.1 Montaje del sistema de medida 28
5.2 Montaje del sistema receptor 28
5.3 Acelerómetro de piezoeléctrico 29
5.4 Montaje de los acelerómetros sobre el excitador 30
5.5 Ensayo con 10 Hz de frecuencia y 100 mV de amplitud 30
5.6 Ensayo con 20Hz de frecuencia y 100mV de amplitud 31
5.7 Ensayo con 20Hz de frecuencia y 200mV de amplitud 31
5.8 Ensayo con 30Hz de frecuencia y 100mV de amplitud 32
5.9 Ensayo con 30Hz de frecuencia y 200mV de amplitud 32
5.10 Ensayo con 40Hz de frecuencia y 100mV de amplitud 33

5.11 Ensayo con 40Hz de frecuencia y 200mV de amplitud 33
5.12 Ensayo con 50Hz de frecuencia y 100mV de amplitud 34
5.13 Ensayo con 50Hz de frecuencia y 200mV de amplitud 34
5.14 Ensayo con excitación Burst random y 100mV de amplitud 35
5.15 Ensayo con excitación Burst random y 200mV de amplitud 35

C.1 Cables de conexión para Arduino 45
C.2 46
C.3 Esquema de conexión del transmisor 46
C.4 Esquema de conexión del receptor 47
C.5 Cambiar Puerto Serie 47

Índice de Tablas

2.1 Precisión del sensor MPU6050 8
2.2 Presupuesto aproximado del sistema 12

4.1 Conexión del acelerómetro MPU6050 con Arduino 20
4.2 Conexión del módulo NRF24L01 con Arduino 20

Índice de Códigos

A.1 Calibración del sensor MPU6050 39
A.2 Código transmisor 40
A.3 Código receptor 42

B.1 Procesado de datos 43

Bibliografía

[1] G. P. Luth, "Chronology and context of the hyatt regency collapse - chronology and context of the hyatt regency collapse by gregory p luth1 abstract this paper | course hero."

[2] R. Gaal, "November 7, 1940: Collapse of the tacoma narrows bridge,"

[3] "Tacoma narrows bridge history."

[4] "A micromachined quartz angular rate sensor for automotive and advanced inertial applications | fierceelectronics."

[5] "Guía imu."

[6] A. Villarroel, G. Zurita, and R. Velarde, "Development of a low-cost vibration measurement system for industrial applications," *Machines*, vol. 7, p. 12, 3 2019.

[7] M. Y. Upadhye, P. B. Borole, and A. K. Sharma, "Real-time wireless vibration monitoring system using labview," pp. 925–928, Institute of Electrical and Electronics Engineers Inc., 7 2015.

[8] C. Chiculita and L. Frangu, "A low-cost car vibration acquisition system," pp. 281–285, Institute of Electrical and Electronics Engineers Inc., 11 2015.

[9] S. Biansoongnern, B. Plungkang, and S. Susuk, "Development of low cost vibration sensor network for early warning system of landslides," vol. 89, pp. 417–420, Elsevier Ltd, 6 2016.

[10] I. M. Díaz, A. Poncela, E. Carrera, F. Miglioretti, M. Petrolo, J. D. Sebastián, A. Escudero, R. Arnanz, I. M. Díaz, and A. Lorenzana, "A low-cost vibration monitoring system for a stress-ribbon footbridge plcs lab view project development of novel damping system for pedestrian structures (reves-p) view project jesús de sebastián cartif a low-cost vibration monitoring system for a stress-ribbon footbridge," 2013.

[11] C. A. A. Basto, "Study on possibilities for low-cost monitoring of historical structures."

[12] "Arduino.cc."

[13] F. V. Fernández, J. María, M. Núñez, and J. O. Granja, "Diseño y control de una prótesis de mano de bajo coste," 2020.

[14] J. Álvaro, F. Carrasco, and C. V. Venegas, "Estudio y diseño de un sistema interfaz cerebro-computador de bajo coste con módulo arduino," 2017.

[15] "Datasheet mpu6000 and mpu6050," 2013.

[16] “Librería i2c.”

[17] “Librería mpu6050.”

[18] Q. Sun, W. Wabinski, and T. Bruns, “Measurement science and technology high-precision and low-cost vibration generator for low-frequency calibration system related content investigation of primary vibration calibration at high frequencies using the homodyne quadrature sine-approximation method: problems and solutions,” 2018.

[19] “Datasheet nrf24l01,” 2006.

[20] “Librería nrf24l01.”

[21] “Putty - a free ssh and telnet client for windows.”

[22] “Matlab - el lenguaje del cálculo técnico - matlab simulink.”

[23] “Características del acelerómetro de piezoeléctrico.”

Glosario

FFT Transformada Rápida de Fourier. 24

I2C Circuito Inter-Integrado. Protocolo de comunicación en serie.. 8

IDE Entorno de Desarrollo Integrado. 3, 13

IMU Unidad de Medidas Inerciales. III, V, 3

LCD Pantalla de cristal líquido. 6

Memoria EEPROM Memoria de solo lectura programable eléctricamente. 7

Memoria RAM Memoria de Acceso Aleatorio. 7

MEMS Sistemas Microelectromecánicos. 2, 3

Offset Valor que define el punto inicial respecto al que se toma una medida. 9

PWM Señal con modulación de pulso. 6

SD Secure Digital. 2, 37

SPI Serial Peripheral Interface. Estándar de comunicaciones utilizado en comunicación en corta distancia.. 10

SSH Secure Shell. 15

TCP Protocolo de Control de Transmisión. 2

TelNet Teletype Network. 15